Aalto University
School of Electrical
Engineering

Pia Lindqvist

Use of a quality monitoring methodology and platform in industrial software development – A feasibility study

Master's Thesis
Espoo 25.01.2016

Supervisor: Professor Arto Visala
Advisors: D.Sc. (Tech.) Juha Itkonen, D.Sc. (Tech.) Timo Lehtinen

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076
AALTO
www.aalto.fi

Abstract of master's thesis

Author PIa Lindqvist

Title of thesis Use of a quality monitoring methodology and platform in industrial software development – A feasibility study

Department Department of Electrical Engineering and Automation

Professorship Automation Technology **Code of professorship** AS-84

Thesissupervisor Professor Arto Visala

Thesis advisors D.Sc. (Tech.) Juha Itkonen, D.Sc. (Tech.) Timo Lehtinen

Date 25.01.2016 **Number of pages** 6+100 **Language** English

Abstract

Ensuring software quality is important for achieving a competitive advantage in the market. Also, it is essential in critical systems, where a computer program cannot fail under any circumstances. In software projects, the best way to accomplish quality is to continuously monitor the state of the product and the development process. This means, that measurements on the software have to be done. However, as it is not adequate to measure everything, the relevant quality characteristics have to be defined. This is very context-specific and should thus be done for each project separately.

In this thesis, the deployment of quality monitoring in software development projects was studied. This included defining a quality model and designing, deploying and using a quality monitoring program. These activities were facilitated by the U-QASAR methodology and platform that provide guidance and support for quality monitoring by measurement data integration.

The study consisted of two parts: a multiple case study and constructive research. The case study was conducted to explore the case project members' experiences on quality monitoring. Observations, interviews and questionnaires were used as research instruments. A constructive research method was used to evaluate the data integration in the U-QASAR platform. Data integration adapters were explored and developers were interviewed and finally a new adapter was implemented. In both parts, thematic analysis was used to process the data.

The case study results showed that quality monitoring can be deployed even in small-scale software projects, but there are certain challenges. For example, it is difficult to connect objective software measurements to more abstract quality objectives. Moreover, practitioners' knowledge on the terminology of software measurement and software quality is not something that can be assumed. However, it is important that the semantic structure is taken care of in order to achieve common understanding. The constructive research revealed needs for further development regarding the data integration architecture of the U-QASAR platform.

Keywords Software development, quality, monitoring, measurement, U-QASAR

Aalto-yliopisto
Sähkötekniikan
korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Pia Lindqvist

Työn nimi Laadunvalvontametodologian ja –järjestelmän käyttö teollisessa sovelluskehityksessä – Soveltuvuustutkimus

Laitos Sähkötekniikka ja automaatio

Professuuri Automaatiotekniikka **Professuurikoodi** AS-84

Työn valvoja Arto Visala

Työn ohjaaja(t) D.Sc. (Tech.) Juha Itkonen, D.Sc. (Tech.) Timo Lehtinen

Päivämäärä 25.01.2016 **Sivumäärä** 6+100 **Kieli** Englanti

Tiivistelmä

Ohjelmistokehityksessä laadunvarmistus on tärkeää, sillä sen avulla voidaan saavuttaa parempi markkina-asema. Ohjelmiston laatu on tärkeää myös kriittisissä systeemeissä, joiden vikaantuminen voi johtaa esimerkiksi mittaviin taloudellisiin menetyksiin. Ohjelmistokehitysprojekteissa laadukas tuote voidaan parhaiten saavuttaa jatkuvasti tarkkailemalla projektin tilaa, mikä edellyttää mittauksia. Kaikkea ei kuitenkaan ole kustannustehokasta mitata, joten on tärkeää määrittää kyseiselle tuotteelle ja projektille tärkeät laatuominaisuudet.

Tässä työssä tutkittiin laadunvalvonnan käyttöönottoa ohjelmistokehitysprojekteissa. Siihen liittyviä aktiviteetteja olivat laatumallin määrittely sekä laadunvalvontaohjelman suunnittelu, käyttöönotto ja käyttö. Toiminnot toteutettiin käyttäen U-QASAR-metodologiaa ja -alustaa, jotka tarjoavat ohjausta ja tukea mittausdataintegraatiolla toteutettavaan laaduntarkkailuun.

Tutkimus koostui kahdesta osiosta: monitapaustutkimuksesta ja konstruktiivisesta tutkimuksesta. Tapaustutkimus tehtiin, jotta laadunvalvonnasta saatavia kokemuksia voitaisiin tutkia. Tutkimusinstrumentteina käytettiin havainnointia, haastatteluja sekä kyselyitä. Konstruktiivisessa tutkimuksessa kohteena oli U-QASAR-alustan dataintegraation toteutus. Jo kehitettyjen integraatioadaptereiden rakennetta tutkittiin ja kehittäjiä haastateltiin. Lopuksi uusi adapteri kehitettiin. Molemmissa tutkimusosioissa temaattista analyysiä käytettiin datan käsittelyyn.

Työn tulokset osoittivat, että laadunvalvontaa voidaan tehdä pienissäkin ohjelmistoprojekteissa, mutta tietyt haasteet on otettava huomioon. Esimerkiksi objektiivisten mittausten yhdistäminen abstraktimpiin laatuominaisuuksiin todettiin vaikeaksi. Lisäksi havaittiin, että laatuterminologian ei voida olettaa olevan hallussa kaikilla projektin sidosryhmillä. On kuitenkin tärkeää, että semanttinen konsensus säilytetään yhteisen ymmärryksen takaamiseksi. Konstruktiivisen tutkimuksen tulokset osoittivat, että U-QASAR-alustan dataintegraatioarkkitehtuuri vaatii jatkokehitystä.

Avainsanat Ohjelmistokehitys, laadunvalvonta, laatu, mittarointi, U-QASAR

Acknowledgements

Writing this thesis has been a heavy but rewarding part of the journey through the university studies and in the process I have grown both as a human and as a researcher. Here I would like to thank those people who have guided and supported me during this project.

First, I would like to thank my thesis advisors, Juha Itkonen and Timo Lehtinen, without whose wise comments and supportive attitude this thesis would not be what it is now. I would also like to thank my supervisor Arto Visala, who kindly accepted a rather different topic for the thesis than it is normally expected in the automation and systems technology department.

During this thesis I had the chance to work with brilliant people from several European countries. Thank you all the members of the U-QASAR project. Especially, thank you Feetu Nyrhinen for patiently guiding me with the programming challenges.

I could not leave the guild of automation and systems technology out of this speech. Especially thank you ASphuksit09 for making me the person that I now am. The adventures with you will always remain in my memories.

The support that I got from home truly gave me the strength to spend all those nights with this work. Thank you Tommi for the endless cheer-up and love that you have showed me in the ups and downs. You have given me the inspiration and determination to do my best in my studies.

Finally, I want to thank my family for their support. Thank you Mom and Dad for the support through the years of education. Your expectations for always doing better made me try harder and demand more from myself. Thank you Lotta for the sisterly peer support, you really understand my thoughts best.

Espoo, 2016

Pia Lindqvist

Table of Contents

1 Introduction

This section presents the motivation to do this study and provides an overview of how this thesis was carried on. First, Section 1.1 provides a background for the study and connects it to a larger context. Then, Section 1.2 presents the research problem and the research questions that were defined to approach the problem. Furthermore, the identified limitations for the study are presented. Finally, Section 1.3 provides an overview of the structure of the thesis.

1.1 Background

Software companies started to show interest towards software quality after the so-called software crisis in 1960s. The term was introduced in conferences organized by the NATO science committee to describe the state of software products at the time: several serious software failures had taken place because of inadequate software development practices and poor software quality. It was realized that software quality is very important in critical systems, where system failure can lead to a significant loss of money and even in losing human life. (Karch, 2011; O'Regan, 2014)

Achieving good quality means iterative work and improvements on the product. Software products are less difficult and less expensive to be changed in the early phase of the development process, especially if the changes concern fundamental parts of the software, such as its architecture (Sommerville, 2011, p. 148). Both processes and products can be improved during development projects. Continuous monitoring of process and product quality creates real-time awareness of the project state. Thus, it helps detecting critical problems in time and enables the decision-making required for changes.

To monitor quality, first it has to be understood what it means. Software quality is a complex concept that builds up from many different elements and highly depends on the context (Jørgensen, 1999). Hence, reasonable quality monitoring requires a project-specific definition of software quality. Based on this definition, the software product and development process can be measured to produce information of the state project.

Although quality has been identified as an important issue, and the research on software quality has been going on for decades, no first-rate comprehensive guides or tools seem to exist for setting up software quality monitoring in an organization. Furthermore, it appears that because of the estimated costs, setting up quality management is rejected in many organizations (Emam, 2005, p. 202) and that open source systems backed up by literature and empirical research seem not to be easily available.

The U-QASAR project, funded by the European Union, intended to develop a solution to monitor software project quality in a flexible and lightweight way. It resulted in a combination of practical guidelines and a supporting tool: the U-QASAR methodology and platform. The methodology and the platform are based on the best practices identified from the literature and they facilitate the creation, deployment and use of software quality monitoring in software development projects.

The U-QASAR methodology provides guidance for defining a context-specific software quality model and for deploying quality monitoring in a software development project. The methodology intends to cover the whole lifecycle of a software product and to support collaborative work practices.

The U-QASAR platform is a browser-based tool that provides means for deploying a quality monitoring program in a software development project. It is based on data integration from the information systems used in the project organization and provides information about the quality of the project on different levels of granularity. The goal is to provide a low-cost solution with personalized user interfaces to support the user experience of different project stakeholders.

1.2 Research problem and research questions

The goal of this thesis is to understand the usefulness and feasibility of the U-QASAR methodology and platform in industrial software development projects. Furthermore, the user experience of the U-QASAR platform is explored to guide the further development. Thus, the research problem of this thesis is "How can a quality monitoring methodology and platform using multi-source data integration be utilized in industrial software development?"

The utilization of the U-QASAR methodology and platform in this thesis means designing a quality monitoring program by the U-QASAR methodology and deploying and using it with the U-QASAR platform. To understand the research objective, the following research questions were set:

RQ 1. Is the U-QASAR methodology perceived as feasible for designing a software quality monitoring program?
RQ 2. Is the deployment of a software quality monitoring program perceived as cost-effective with the U-QASAR platform?
RQ 3. Is a software quality monitoring program deployed with the U-QASAR platform perceived as useful for monitoring software quality?
RQ 4. What improvement factors can be identified for the U-QASAR platform?

To answer these research questions, two qualitative research methods were used in this study. *A multiple case study* was conducted to explore the use of the U-QASAR methodology and platform in real software development projects. Additionally, *a constructive research approach* was used to explore the workload of implementing data integration between the U-QASAR platform and a software measurement tool.

Several data collection instruments were used for each of the research questions. An overview of these is presented in Table 1. The research methods, the data collection instruments and the data analysis are described more comprehensively in Section 4.

The limitations of the research methods to answer each of the research questions are presented in Table 2. In addition to these research question specific limitations, a main limitation for this study was the contribution of only one researcher in the interpretation and analysis of the collected raw data.

Table 1. Data collection instruments used to answer the research questions.

Instrument	RQ 1	RQ 2	RQ 3	RQ 4
Pre-questionnaire				
Post-questionnaire	x			
Background information form				
Follow-up questionnaire			x	x
Final questionnaire			x	x
Follow-up interview 1			x	x
Follow-up interview 2			x	x
Follow-up interview 3			x	x
Final interview		x	x	
Workshop discussions	x			
Indicator evaluations	x			
Adapter diary		x		x

Table 2. Limitations of the study for each research question.

Research question	Limitations of the research methods
RQ 1. Is the U-QASAR methodology perceived as feasible for designing a software quality monitoring program?	The collected data is limited to three case projects of which one is a pilot project on the U-QASAR concept itself. These projects are presented in Section 4.1.
RQ 2. Is the deployment of a software quality monitoring program perceived as cost-effective with the U-QASAR platform?	The collected data is limited to the deployment experiences of two case projects, the analysis on deployed monitoring programs in two other cases and finally the studied effort in linking the U-QASAR platform technologically to the project environment.
RQ 3. Is a software quality monitoring program deployed with the U-QASAR platform perceived as useful for monitoring software quality?	The collected data is limited to the experiences on quality monitoring by four case studies. Two of these projects were able to do proper quality monitoring with the platform but two of them based their experiences on opinions.
RQ 4. What improvement factors can be identified for the U-QASAR platform?	The collected data has the same limitations as with RQ 3, but was additionally extended by constructive research.

1.3 Structure of the study

This thesis is divided to eight sections, where the first section is the introductive part to the study. To describe the essential concepts and theory as well as prior research on measuring software and monitoring software quality, Section 2 provides a literature review on the software quality and software measurement. In Section 3, the U-QASAR methodology and platform are described to provide an overview of the means of implementing software quality monitoring in this study. The research methods used in this study are presented in Section 4.

Sections 5 and 6 present the results that were collected by the research instruments. In Section 7 the research questions are answered and the implications of the findings are discussed. Furthermore, the validity and reliability of this study are discussed. Finally, Section 8 presents a summary of this work and provides suggestions on future work on this topic.

The contribution of the literature review and the research methods for each research question is presented in Table 3. The results from the multiple case study were used to answer each of the research questions and the results from the constructive research contributed in the questions concerning the U-QASAR platform.

Table 3. Thesis contents' contribution to the research questions.

Research question	Research method	Related sections in the literature review	Related sections in the results
RQ 1.	Case study	2.1.1 Monitoring software quality 2.1.2 Modeling software quality 2.1.3 Defining software quality 2.2.1 Goal-oriented software measurement 2.2.2 Success factors in deploying software measurement programs	5.1 The usefulness of the U-QASAR methodology 5.2 The outcome of the U-QASAR methodology
RQ 2.	Case study Constructive research	2.2.1 Goal-oriented software measurement 2.2.2 Success factors in deploying software measurement programs	5.3.1 Deployment of a quality monitoring program
RQ 3.	Case study	2.2.1 Goal-oriented software measurement 2.2.2 Success factors in deploying software measurement programs 2.2.3 Quality monitoring by automated software measurement	5.3.2 Quality monitoring 6 Results from the constructive research
RQ 4.	Case study Constructive research	2.2.3 Quality monitoring by automated software measurement	5.3.3 Technical features 6 Results from the constructive research

2 Literature review

This literature review was conducted to characterize the concepts behind the research questions and to provide an overview of the research done on software quality and software measurement. As designing a quality monitoring program (RQ1) requires defining a project-specific quality model, Section 2.1 explains how quality can be modeled and defined. A quality monitoring program is a modification of a software measurement program, and thus Section 2.2 provides a view on measuring software and deploying a software measurement program (RQ2, RQ3). Furthermore, Section 2.2 presents tools that the literature provides for using a software measurement program (RQ4). Finally, Section 2.3 summarizes the literature review and briefly describes a gap identified in the prior research.

This literature review was carried out by collecting and reading books and research papers related to software quality assurance and software measurement. To increase the credibility of the study, the papers were retrieved from scientifically appropriate databases, such as the database of the Institute of Electrical and Electronics Engineers (IEEE) and ScienceDirect. Relevant publications were searched from Scopus with appropriate search strings. Examples of the search strings in Scopus are the following:

(TITLE-ABS-KEY ("software development") AND TITLE-ABS-KEY (model*) AND TITLE-ABS-KEY (monitor*) AND TITLE-ABS-KEY (quality)) and

(TITLE-ABS-KEY ("software measurement program") AND TITLE-ABS-KEY (quality)).

Backward snowballing, i.e. searching references of relevant papers was used, which lead to a wide collection of papers. Furthermore, publications were searched directly from the databases or the Google search engine with the name of the publication. Google was also used to find terminology for the search strings from unscientific publications such as blogs and Wikipedia articles. Examples of the keywords used in Google included “quality monitoring” and “software measurement program”. New keywords were found during the study as the knowledge of the topic and the vocabulary increased.

2.1 Concept of software quality

This section presents the fundamental ideas of defining and modeling software quality. Section 2.1.1 first provides an overview of the attempts to model software quality in literature and then Section 2.1.2 presents different viewpoints and approaches to define software quality in a certain context. Finally, Section 2.1.3 presents the connection between software measurement and quality improvement.

2.1.1 Monitoring software quality

Software quality is a very abstract and subjective concept. However, due to its financial importance, it should be ensured in software products. Thus, software quality needs to be monitored and objective measurements are required. According to Jørgensen (1999), the concept of quality measurement is misleading. Instead, he presents that quality monitoring is based on linking the subject of quality to the features of software products

and processes. Thus, it can be said that *software measurement* is a part of quality monitoring. Furthermore, q*uality models* can be used to illustrate the linking between the concept of quality and the features of software products and processes.

Although the literature identifies the concept of *a quality measurement program* for software projects, in this thesis the term *software quality monitoring program* or *quality monitoring program* is preferred to highlight the difference between measuring software and monitoring quality. A quality monitoring program is in this thesis defined as a combination of a quality model, identified data sources for the required measurements, defined calculations for creating the values of the elements in the quality model and finally defined threshold values to indicate the state of the quality of these elements. The quality monitoring program can be seen as a special edition of a *software measurement program* that is described in Section 2.2.1.

2.1.2 Modeling software quality

Quality models (see Figure 1) present the features of the software products and processes as a composition from the most abstract level to the concrete measurements that can be made on the software product or process. The terminology related to quality models seems not to have been stabilized in literature and the elements in the models have varying titles.[1] In this thesis, a quality model consists of three levels of abstraction: *quality objectives*, *quality indicators* and *quality metrics*. Quality objectives construct the highest abstraction level of the quality model. Quality objectives are defined by quality indicators, which create the second level of abstraction in the quality model. On the lowest level, the quality metrics present the raw measurement data.

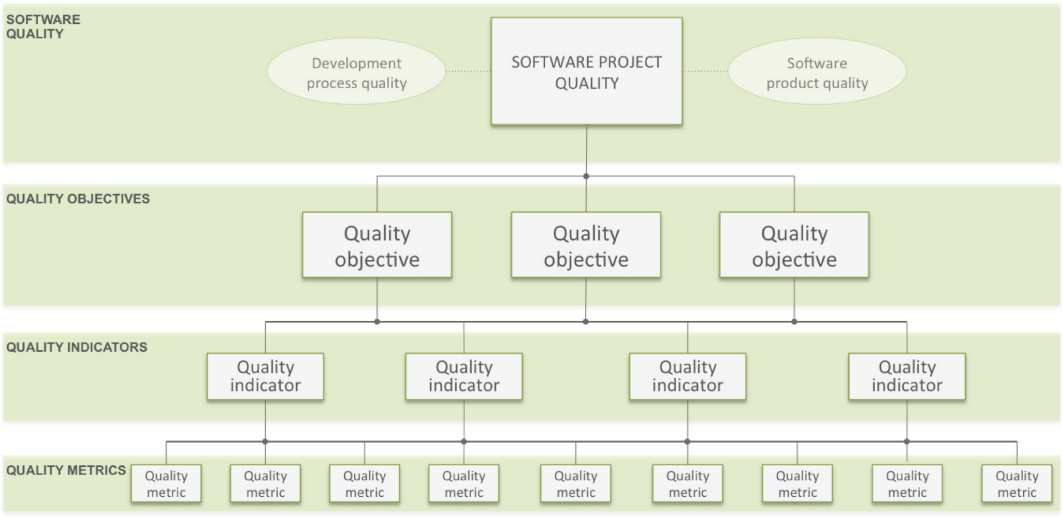

Figure 1. A quality model consists of quality objectives, quality indicators and quality metrics.

[1] In literature, quality objectives are alternatively called ”characteristics”, (Jørgensen, 1999) ”factors” (Fenton, 1994) or ”goals” (Basili and Weiss, 1984), quality indicators are presented as ”questions” (Basili and Weiss, 1984) or ”criteria” (Fenton, 1994).

Since the late 1970s, there have been several attempts to develop a universal model for software quality, see for example (McCall et al., 1977), (Boehm, 1978), (Garvin, 1984) and (Dromey, 1995). Quality models have even been attempted to be standardized by The International Organization for Standardization ISO (2010, 2001) and IEEE (1998). The Consortium for IT Software Quality (CISQ) (2012) has also provided a specification for a limited part of software quality. To provide a universal view, the definitions and standards try to cover all of the possible elements in the quality model.

The quality models proposed in literature mainly include product-based objectives, but also the quality of software processes should be taken into account in monitoring software quality. Dowson (1993) presented that process improvement would lead to higher product quality. In his book, Sommerville (2011) states that dependable processes lead to dependable systems. Wagner (2013) has extrapolated this to as "high-quality processes also lead to high-quality products".

The proposed quality models have been under critique for several reasons. Hofman (2011) states that there is no commonly accepted model for software quality. Deissenboeck et al. (2009) have identified from literature the following weaknesses for the proposed quality models:

- Quality models do not conform to an explicit metamodel, which has led to vague description of the model elements
- Quality models do not address different views to quality
- Quality models lack decomposition criteria
- Quality model elements are overlapping
- Quality model frameworks do not provide instructions to use the model
- Quality models do not specify the impact that one element has on the quality
- Metrics in quality models do not have clear validation
- Metrics in quality models do not respect the measurement theory

Despite the various attempts to model software quality and the sense-making contents of the proposed models, it has been claimed that they are largely based on speculation and hypotheses, without being scientifically confirmed (Dromey, 1998). Moreover, the standards have been found to be inadequate and inaccurate. Al-Kilidar (2005) reported a number of problems with the ISO/IEC 9126 standard. These were for example ambiguity and overlapping in some concepts, ignorance of some important characteristics and unachievable measurements.

2.1.3 Defining software quality

Remarkable in the existing quality models is, that not all of the elements of the models need to be present in one software product and a suitable set of elements should be selected for a certain project by some criteria, i.e. to *define the quality* in the project. The definition of quality is a part of larger concept, a quality monitoring program, and many frameworks have been developed to facilitate it. These will be discussed in Section 2.2.1 and this section will focus on describing different viewpoints to quality.

In despite that a great number of definitions on software quality and software metrics have been proposed in literature and the models have even been standardized, it seems that there still is no widely accepted universal definition that could be applied in every project. Instead, organizations have to start their definition process merely from the beginning. Jørgensen (1999) proposes that the problem is actually in standardizing how software quality is interpreted. He also predicts that such consensus will never be achieved.

Defining software quality is not simple, as many factors contribute to what quality actually means. The main reason for this is the uniqueness of software projects: they highly depend on the people in the project team, the product that is being developed and the context where the developed product will be used. Also the intangible nature of software makes the definition challenging, as it means assessing immaterial products of high complexity. Furthermore, many factors creating the quality can only be assessed subjectively (Pressman, 2010, p. 402), whereas the definitions should be objective in order to be able to measure them.

A look on the proposed quality models also indicates that the connecting the metrics to the higher-level elements is difficult. The quality models provided in literature do not provide more complex calculation formulas than dividing existing features with required ones (see for example ISO/IEC, 2010). Furthermore, based on the research on this literature review, no extensive collection of metrics connected to quality objectives is available. This implies that the connections between quality model elements are difficult to construct.

Garvin (1984) presented the five closely related approaches to product quality (see Table 4). They suggest that there are different needs for quality information among project stakeholders. These approaches apply also on software quality and should be taken into account when it is defined (Dromey, 1998; Kitchenham and Pfleeger, 1996).

Table 4. Approaches to product quality by Garvin (1984) and descriptions by Pressman (2010).

Approach	Description	Relevance
Transcendental approach	"Quality is something that you immediately recognize, but cannot explicitly define."	Understanding and defining the quality of a software product is not easy and requires good guidelines and practices.
User-based approach	"Quality in terms of an end user's specific goals."	When defining the quality of a software product, customers' differences should be kept in mind.
Manufacturing approach	"Quality in the terms of the original specification of the product."	The product should also reflect its specification in order to be of good quality.
Product-based approach	"Quality can be tied to inherent characteristics of a product."	The product constructs its own quality.
Value-based approach	"Quality based on how much a customer is willing to pay for a product."	The customers and the market should be kept in mind in decision-making.

The transcendental approach means the feeling of the presence of the quality and it is perhaps the most difficult dimension to measure or even describe. The product-based approach stands for the features of the product itself and their role in creating quality. The value-based approach is for the marketing of the product and the price that it is offered for. If the customers see that the price of the product meets the product's value, they are willing to buy it. This approach is often excluded in the quality definition while it is very important nowadays (Deissenboeck et al., 2009 based on Kitchenham and Pfleeger, 1996; Wagner, 2007).

The user-based approach is very important, especially when developing software for the market. It means the users experience on the product and finding all their desired features in it. If the specification of the product would be perfect and the product would meet it, it should mean that the user-based quality has been achieved. However, in practice it is difficult to write a specification as detailed that it would include all the requirements of the end-users and operating environments. The manufacturing approach means meeting the specification without any other dependencies.

2.2 Software measurement

As software measurement is a part of the software quality monitoring program, it is justified to introduce it here. This section presents the fundamental ideas of measuring software and creating a software measurement program. Section 2.2.1 presents the concept of a software measurement program and the idea of goal-oriented (top-down) measurement. Section 2.2.2 presents a literature review of success factors for deploying a software measurement program and Section 2.2.3 introduces automated software measurement and tools developed for it.

> *"Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way so as to describe them according to clearly defined rules." –Norman Fenton, 1995.*

Measuring software increases the knowledge of the development activities and hence is supposed to ease the management of software development project. Van Solingen and Berghout (2001) have in their literature review listed the following examples of what measurement results have caused:

- Increased understanding of software development processes.
- Increased control of the software development process.
- Increased capacity to improve the software development process.
- More accurate estimates of software project costs and schedule.
- More objective evaluations of changes in technique, tool, or methods.
- More accurate estimates of the effects of changes on project cost and schedule.
- Decreased development costs due to increased productivity and efficiency.
- Decrease of project cycle time due to increased productivity and efficiency.
- Improved customer satisfaction and confidence due to higher product quality.

Moreover, Tahir and Jafar (2011) have identified from literature the following areas where measurement data is used: 1) Monitor and control, 2) Decision making, 3) Software process improvement, 4) Performance improvement and 5) Organization health. All of these aspects contribute in enhancing the software quality and emphasize the idea that software quality monitoring and improvement are based on software measurement. To monitor software quality, the measurement data for the metrics has to be produced or collected and then processed and analyzed. The results need to be stored so that they can be used in trends that present the historical state of the measured objective and the project.

Many frameworks have been proposed for designing a measurement program for software development. In his book, Southekal (2014) presents the timeline of frameworks proposed for software measurement and their allocation to organizational and measurement frameworks. Despite the high number of frameworks, Southekal states that the most common way to do software measurement is the goal-oriented approach.

2.2.1 Goal-oriented software measurement

Software measurement means tracking different features of a software product or a software process, i.e. the elements of a software quality model. The set of necessary subjects required for software measurement is called a software measurement program. A software measurement program includes a quality model, defined threshold limits for the quality model elements to indicate good and bad quality and the identified sources for the measurement data.

The quality model can be defined by two approaches: by the top-down approach or by the bottom-up approach (O'Regan, 2014). In the former, the goals of the measurement, i.e. the quality objectives, are defined first and then the metrics for them are identified. The latter approach is reversed (Parkinson et al., 2010). It seems that the top-down approach is more popular as more research was found on it.

Tahir and Jafar (2011) argue that implementing a measurement program requires both of these approaches: a top-down approach to cope with the problems of the bottom-up approach. The necessity of the top-down approach springs from the better targeting of the measurements by knowing what is important to be measured and the possibility to align the measurements with the business goals of the organization. However, the analysis of the measurement results should be started from the metric layer to find that first of all if the collected data is valid and then whether it describes the related quality indicators, and that the indicators describe the quality objectives in the correct way.

The top-down approach to software measurement is called goal-oriented software measurement (Van Latum et al., 1998). It tries to first understand the important matters in the larger context (goals) and then define appropriate measures to them (Rombach and Ulery, 1989). It is suggested that the goals are explicit and clearly defined (Briand et al., 1996) to be easy to understand and interpret.

Briand (1996) also lists the following advantages of the goal-driven measurement approach:

- Ensuring adequacy, consistency and completeness of a measurement plan and therefore of the data collection
- Helping to manage complexity of the measurement program
- Promoting consensus about measurement and improvement goals
- Stimulating structured discussion about measurements

A number of different frameworks have been developed for the goal-oriented software measurement approach. Some of them are presented in Table 5. The best-known approaches are the goal/question/metric (GQM) approach, the AMI approach, the balanced scorecard (BSC) and the practical software and systems measurement (PSM).

Table 5. Frameworks for goal-oriented software measurement.

Framework	Abbr.	Year	Reference
Goal/Question/Metric	GQM	1984	(Basili and Weiss, 1984)
Application of Metrics in Industry	AMI	1992	(Rowe and Whitty, 1993)
Balanced scorecard	BSC	1992	(Kaplan and Norton, 1992)
Goal/Question/Indicator/Metric	GQ(I)M	1996	(Park et al., 1996)
Model, measure, manage paradigm	M^3P	1997	(Offen and Jeffery, 1997)
Measurement Program Survey Package	MPSP	2000	(Berry and Jeffery, 2000)
meta-Measurement Project	M2P	2001	(Berry and Vandenbroek, 2001)
Nokiaway	-	2001	(Kilpi, 2001)
Practical software and systems measurement	PSM	2006	(US Dept. of Defense and US Army, 2006)
MIS-PyME	-	2007	(Münch and Abrahamsson, 2007)
GQM+Strategies®	-	2007	(Basili et al., 2007)
Optimum measures set decision	OMSD	2009	(Bhatti et al., 2009)
Practitioner-based measurement	PBM	2010	(Parkinson et al., 2010)
Structured Prioritized Goal Question Metrics	SPGQM	2010	(Tahir and Gencel, 2010)

The GQM approach is directed for development projects to define what should be measured in the products and processes and it has achieved the status of a measurement paradigm. Many of the other frameworks are based on the GQM approach and have been developed to overcome the problems with the original framework (Tahir and Jafar, 2011). The BSC approach focuses more on the business side and the business goal

alignment with the measurements (O'Regan, 2014) and the PSM approach mainly defines the activities of the measurement process (Asato et al., 2009).

Based on the literature, by far the most used approach to design a software measurement program is the GQM approach and its variants. GQM proposes a top-down approach for defining quality and a bottom-up approach for analyzing and interpreting the resulting data (Van Solingen and Berghout, 2001).

Originally, the GQM method provides the following six steps for implementing software measurement (Basili and Weiss, 1984):

1. Establish the goals of the data collection
2. Develop a list of questions of interest
3. Establish data categories
4. Design and test a data collection form
5. Collect and validate data
6. Analyze data

A more modern approach to GQM by Fuggetta et al. (1998) is presented in Figure 2. The initial step to be taken is exploring and understanding the context in which the software is being developed. Then, the desired goals to be monitored are identified and a GQM plan – which in this work is called a quality model – is defined. On the base of the quality model, a measurement plan – which in this work is called a measurement program – is designed. The next step is to deploy the measurement program and start the data collection and validation for the metrics. Finally, the experiences and materials are archived for reuse purposes. (Fuggetta et al., 1998).

Figure 2. The GQM activities and outcomes by Fuggetta et al. (1998)

2.2.2 Success factors in deploying software measurement programs

According to Briand (Briand et al., 1996), there are some issues concerning the measurement acceptance and reliability that have to be addressed. They are:

- Getting commitment is not a matter of course
- Training the personnel to understand measurements
- Adapting tools might be a considerable effort

Briand also offers solution propositions for these issues. The commitment can be gained by involvement of the personnel in the whole lifecycle of the measurement program. This includes goal setting, measurement planning, data collection and interpreting the data. The training of the personnel should include the purposes of the measurements, the roles of the project personnel in data collection, avoiding reliability issues in the data collection and tool training for the adequate use of them.

Parkinson (2008) states based on Dekkers and McQuaid (2002) that "anecdotal evidence on the success of metrics programs suggests that 75%-80% or more of metrics programs fail to deliver their objectives", which is a very high number. Börjesson (2006) has identified from literature the following problems in measuring software and data gathering:

- It is expensive and time-consuming
- It affects the busiest people
- It can be perceived threatening
- There can be considerable confusion on
    - What data is gathered
    - How the data is used

Despite these hinderers to software measurement, it is still seen that it is necessary to measure software development (Börjesson, 2006; Grady, 1992). Table 6 presents success factors and lessons learned that were mentioned in the papers by Alandes et al. (2012), Tahir and Jafar (2011), Parkinson et al. (2010), Coman et al. (2009), Frederiksen and Iversen (2003), Gopal (2002) and Niessink and Van Vliet (2001). These papers have approached the subject by a literature review, by a case study or both. The success factors can roughly be categorized under four topics: the defined quality model and the collected data, the people, the practices and the data collection and tools. The data collection and tools are discussed in Section 2.2.3.

The quality model used in the measurement program should be well defined, efficient, relevant to the practitioners and not too large. The collected measurement data should be unified and properly used in the improvement processes, instead of just collecting it. It is also important that the practitioners understand the collected data. The privacy of the data should be respected and it should not be used in management decisions against the practitioners. The practitioners should have full access to their own data.

The commitment of the practitioners is one of the key factors to software measurement success (Parkinson et al., 2010) and it has also been found that developer involvement in defining and using the measurement program is valuable. The practitioners should be

trained well to understand the measurement program and the collected data, and some even suggest that internal champions and external measurement gurus should be used in the deployment of program.

Some practices have been found helpful in deploying a software measurement program in a company. Starting from a small set of objectives and metrics and incrementally developing and constantly improving the program has been found useful, as well as the reuse of the metrics, which eliminates the need to define the measurements from the beginning every time that a program is wanted to be deployed.

Table 6. Success factors in deploying a software measurement program mentioned in (Alandes et al., 2012), (Tahir and Jafar, 2011), (Parkinson et al., 2010), (Coman et al., 2009), (Frederiksen and Iversen, 2003), (Gopal et al., 2002) and (Niessink and Van Vliet, 2001)..

Category	Factor
QUALITY MODEL AND COLLECTED DATA	Well defined and useful quality model Business goal alignment Not too large quality model Careful data analysis Understanding the collected data Efficiency of quality model and tools Integrity of data Proper use of the measurement data Practitioners full access to their own data Data privacy
PEOPLE	Practitioner commitment Manager commitment Developer involvement Practitioner training Internal metrics champions External metrics gurus
PRACTICES	Incremental approach Constant improvement of the program Reuse of metrics Communication & feedback Transparency of measurement process Implementing changes based on the data Monitoring the implemented changes Measurement incentives
DATA COLLECTION	Well defined data collection Automated data collection Lightweight data collection Data collection with high frequency Data integration from existing tools
MEASUREMENT TOOLS	Tool support Summaries of data Customized views Measures repository Self-monitoring tool

The measurement process should be transparent, so that all the stakeholders know what is being measured and possible flaws and errors can be reported. Some also suggest incentives to be used to make the practitioners more committed to the measurement. As mentioned, the data should be used properly and changes should be implemented on the base of it. These changes should also be monitored.

Although the presented criteria have been documented in scientific publications, Frederiksen and Iversen (2003) found in their literature survey that actually many of the proposed success factors were broad and difficult to confirm and would need more experimental studies to be proved.

2.2.3 Quality monitoring by automated software measurement

Many different aspects of software can be measured, which has caused a high number of different measurement tools to emerge (Deissenboeck et al., 2008). As mentioned in Section 2.2.2, one of the success factors for a measurement program is automated metric data collection. One way to implement this is measurement data integration from the existing tools. *Automated software measurement* combined with automated analysis and visualization of the measurement data allows lightweight quality monitoring.

Numerous tools have been proposed for automating software measurement by data integration. In this work, it is assumed that the tools still relevant today have been developed within approximately the past ten years. Furthermore, the studied tools were limited to non-commercial measurement tools. Table 7 presents a brief summary of the tools that were found by searches in Scopus and snowballing with the aforementioned limitations. A more comprehensive table of the tool features can be seen in Table 23 in Appendix A.

Table 7. Tools for measurement data integration and quality control. An empty field means that the information was not clearly available.

Name	Year	Focus	Quality modeling	Reference
SQuAVisiT	2007	Quality control		(Roubtsov et al., 2007)
ConQAT	2008	Quality control		(Deissenboeck et al., 2008)
Alitheia Core	2009	Quality evaluation	Yes	(Gousios and Spinellis, 2009)
SPDW+	2010	Quality metric capturing		(Silveira et al., 2010)
SOFAS	2011	Analysis result integration		(Ghezzi and Gall, 2011)
3C	2012	Quality measurement		(Janus et al., 2012)
QualitySpy	2012	Unified data collection		(Jureczko and Magott, 2012)
Dione	2012	Defect prediction		(Caglayan et al., 2012)
UQM	2012	Quality monitoring	Yes	(Schrettner et al., 2012)
ASSIST	2013	Software measurement	Yes	(Keser et al., 2013)
DePress	2014	Defect prediction		(Madeyski and Majchrzak, 2014)
SQA-Mashup	2014	Quality data integration		(Brandtner et al., 2014)
U-QASAR	2015	Quality modeling	Yes	-

For only seven out of thirteen tools, a website or an online repository could be found by using a search engine. Six of these were available for downloading and installation, and additionally contact information was provided for one tool. For the rest, only scientific articles could be found. Furthermore, only one or two articles for each of the tools could be found.

It is notable that only few of the tools were found to support quality modeling and platform customization. Only two of the tools were found to support both of these. Furthermore, only one of the tools was found to provide statistical support such as integration of R or Matlab. None of the tools provided a support for decision support e.g. by integration of a Decision Support System (DSS). Moreover, many of the tools presented in literature are developed in a relatively small research setting or as an in-house tool for internal quality assurance and software measurement.

The literature provides many studies on the requirements and success factors for software measurement tools. Madeyski and Majchrzak (2014) explored the literature on automated measurement tools as a part of their work on developing a software measurement and defect prediction framework. They found that many of the available tools only collect data but do not support predictive modeling. Another finding was that only a small part of the tools provide a good solution for measurement data integration and the most of the tools are developed only for internal use of the developer institution. Based on a literature review, they identified the following requirements for their platform:

- Workflow visualization and usability
- Support for collaboration
- Extensibility
- Standalone implementation
- Export/import support
- Ready for commercial use
- Open source
- Language and technology independent

The requirement "standalone implementation" is questionable, as it is only based on beliefs of the researchers that a web-based tool would not be adapted by organizations because of data safety reasons. Only two of the developed tools presented in Table 7 are standalone implementations. This work does not identify a standalone implementation to be a requirement for a data-integration platform.

Brandtner et al. (2014) have also explored the requirements for a data integration platform for the needs of continuous measurement. As a conclusion of a literature review, they listed the following requirements as the most important for their implementation:

- Ability to locate quality hot-spots in source code
- Dynamic arrangement of information shown in the user interface
- Providing awareness of the activities of co-workers
- Ability to discover changes in real time

- Interactive visualization of a person's role
- Ability to interoperate with other software engineering tools
- Independent development of the user interfaces

In Table 6 were presented the findings on the success factors concerning the data collection and tool support. The data collection should be well defined, lightweight and automated as well as possible. It is also stated that the data collection should be done by integration from existing measurement systems. Also, the collection should be done with high frequency. Tool support should be provided for analyzing, monitoring and storing the data. Summaries and aggregations should be provided as well as customized views to different stakeholders. The tools should be self-monitoring and support a repository for the measures.

Fonseca et al. (2015) have explored the existing data integration tools for measurement support by a mapping study. They found the following gaps in the tool landscape:

- Lack of concern with semantics
- Limited coverage with respect to the measurement process or the measure categories addressed by the integrated tool suite
- Lack of alignment to quality-related standards and maturity models
- Failure to consider a holist view of the (software) process, leading to absence of integration in the process layer

2.3 Summary and gap in prior studies

In this section, the concepts behind the research questions were characterized and a review of the prior literature on software quality and software measurement was provided. Concerning research question 1, it was understood that software quality can be modeled with decompositional quality models and defined by identifying the context-specific important features in software processes and products. Concerning research questions 2 and 3, goal-oriented software measurement and success factors in the deployment of a software measurement program were presented. Finally, literature on automated software measurement and tools for implementing it was presented to better understand the research question 4.

Many methodologies have been developed for defining quality models, and many researchers have further developed the best approaches. However, the tool support for measurement and quality improvement seems to be quite scattered and the U-QASAR project has identified a gap in the literature for an efficient and customizable data-integration tool for modeling and monitoring software product and process quality. The literature review in this thesis supports the gap, as can be deduced from Table 23 in Appendix A. The U-QASAR methodology and platform presented in Section 3 tries to fill this gap and provide a methodology combined with a platform tool to facilitate the software product and process quality modeling needs as well as daily monitoring of these matters by different stakeholders.

3 The U-QASAR methodology and platform

The U-QASAR methodology (see http://webbook.uqasar.eu/) was developed to support the design, use and improvement of a software quality monitoring program. To enable the use of the methodology, a related platform was developed for deploying and using the quality monitoring program. The methodology intends to cover the quality monitoring during the whole lifecycle of a software product and it includes the following four parts:

1. Quality monitoring program design
2. Daily software development control
3. Process improvement
4. Quality monitoring program evolution

This thesis focuses on designing a quality monitoring program and deploying and using it with the U-QASAR platform. This section describes the U-QASAR methodology and platform in the way that they were used in the case study in this thesis. First, Section 3.1 describes the means that the U-QASAR methodology provides for defining a quality model and then Section 3.2 describes the idea and implementation of the U-QASAR platform.

3.1 The U-QASAR methodology

This section describes the first and the second part of the U-QASAR methodology that were explored in this study. First, Section 3.1.1 presents the method for defining a quality model and then Section 3.1.2 describes how a quality monitoring program is designed on the basis of the defined quality model.

3.1.1 Defining a quality model: the Quality Objective Setting method

As described in Section 2.1.3, when an institution wants to monitor software quality, they first have to define what quality means for that particular institution. For defining quality, the U-QASAR methodology provides the Quality Objective Setting (QOS) method, which is intended to be a lightweight, collaborative practice and is based on the GQM approach. The QOS method can be used to define the quality without prior quality definition or quality monitoring program, but existing knowledge can be used as supporting material.

The QOS method suggests a workshop setting, in which different stakeholders of the project or the personnel of the institution's different business units and areas of expertise can collaboratively define a quality model. The result from using the method includes the defined quality model for the project or institution and an initial outline for implementing the quality monitoring program.

The work phases of the QOS method are presented in Figure 3. In the preparative step, the practitioners are provided with basic information and examples of defining software quality. Then, the quality objectives are defined by brainstorming, which is followed by categorization, presentation and prioritization of them. This creates understanding of the most important quality objectives that should be regarded in future.

When the quality objectives are established, they are further elaborated. This means defining the quality indicators and metrics for them as well as determining initial threshold limits to indicate the state of the quality components. All of the objectives can be elaborated or only a most highly prioritized part of them. In U-QASAR, a template is provided to facilitate the quality objective elaboration (see Appendix A).

Also, the indicators are evaluated by prioritization. This is done to understand the indicators more thoroughly and to raise discussion of their feasibility and suitability. After the definition phase, a final discussion is conducted with all of the participants using the QOS method. The discussion can include topics such as the coverage of the defined quality model, the design and deployment of a quality monitoring program and the other further actions.

In Figure 3 the QOS method structure is presented as a continuous process. This means, that the quality model can and should be revised during their lifecycle. Furthermore, existing models can be used as the base of the definition of the quality model for new projects, which allows the reuse of previous attempts and enhances continuous learning.

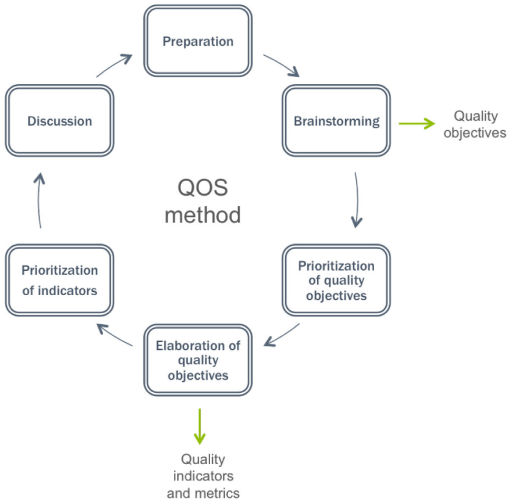

Figure 3. Work phases of the quality objective setting method.

3.1.2 Designing a quality monitoring program

A quality monitoring program includes a refined quality model with the desired quality components. Moreover, it defines the threshold limits for all of the quality components as well as the data sources for the metrics. The designed program can be directly entered to the U-QASAR platform. Summarizing, a quality monitoring program can be designed by taking the following actions on the defined quality model:

1. Refining the model and finding data sources
    i. Deciding which parts of the quality model will be implemented
    ii. Finding out what can be measured automatically
    iii. Finding out what can be measured manually
2. Implementing the data collection instruments
3. Defining the formulas for calculating the values for the overall quality, the quality objectives and the quality indicators
4. Defining the limits for the quality elements

The structure of the design process is presented in Figure 4. First, all the loose components are removed from the defined quality model and the data sources for the defined metrics are identified. This step can reveal that everything in the defined quality model cannot be measured. There can be different reasons to leave some quality model elements out from the quality monitoring program. For instance, it can be found that the data for the metric or the metrics is impossible or too laborious to collect.

The second step is to create the instruments for collecting the metric data either manually or automatically. These include for example questionnaires for the manually collected data and data integration adapters for automatic data collection by data integration with the U-QASAR platform. The effort of implementing a data integration adapter for the U-QASAR platform is studied in this thesis and is further discussed as a part of the results in Section 6.

Finally, it has to be defined how the values for the quality objectives and indicators are calculated and what the values of quality model elements mean, i.e. setting the upper and lower threshold limits for them. The practitioners can decide the limits by themselves or accordingly to recommendations in the institution policies. The formulas for calculation have to be validated after creating them to ensure correct results.

Similarly to the quality model definition process, the quality monitoring program design process is presented as continuous in Figure 4. This means, that the monitoring program should be revised during the use of it and the appropriate changes to it should be done as the project proceeds and the needs for the information for quality monitoring change.

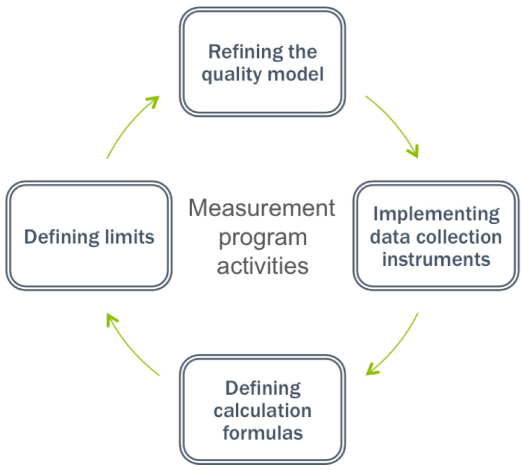

Figure 4. Designing a quality monitoring program.

3.2 The U-QASAR platform

This section presents the U-QASAR platform, which was developed to facilitate modeling and monitoring software quality and to implement quality metric data collection. First, Section 3.2.1 presents the measurement data collection in the U-QASAR platform and then Section 3.2.2 presents the graphical monitoring features.

3.2.1 Data collection by system integration

The basic concept of the platform is that it does not itself analyze source code and generate measurement data but retrieves it from other information systems that measure and produce project-related information. Alternatively, the platform accepts user inputs. According to the defined formulas, the platform calculates the values for the quality indicators and objectives.

The data collection from other information systems is enabled in U-QASAR by data integration adapters, which are relatively small Java-based interfaces implementing the connection between the U-QASAR platform and the target systems. The basic working principle of a data integration adapter is presented in Figure 5. The U-QASAR platform can query the target system and the adapter converts the available data to a format that is supported by the U-QASAR platform interface. The query can be directed to for example a database of the target system or an *application programming interface* (API), if the target system implements one.

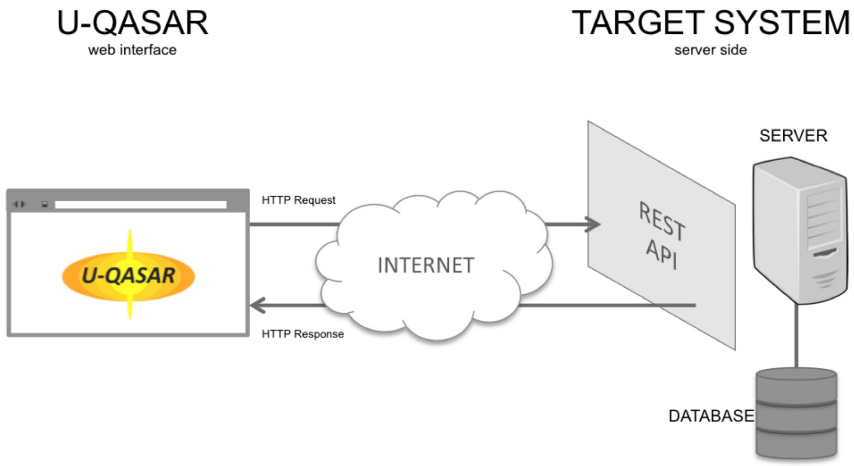

Figure 5. Operating principles of the U-QASAR data integration adapter.

To enable the mapping of the data, the U-QASAR platform defines a generic U-QASAR adapter interface, which can be implemented by the data integration adapters. As presented in the U-QASAR platform design document, the interface includes the following two methods:

```
public List<Measurement> query(BindedSystem bindedSystem, User user,
        QueryExpression queryexpression) throws uQasarException;
public List<Measurement> query(String bindedSystemURL,
        String credentials, String queryExpression) throws uQasarException;
```

The following were defined in the method code:

- bindedSystem/bindedSystemURL as URL to the system to connect with
- user/credentials as the identification means for the system to connect with
- queryExpression as the query to be executed.

The return value of the methods is a object list including the values of the queried measurements. Each of the returned object in the list is a key-value pair where the key is the name of the metric and the value is the value of the measurement:

```
public class Measurement {
    private uQasarMetric metric;
    private String measurement;
    // Other content stripped, e.g. getters, setters, toString() }
```

3.2.2 Quality monitoring

The U-QASAR platform provides means to graphically model and monitor software project quality. As presented in Figure 6, four different views are provided for monitoring the quality. First, there is an overview page for the project, which shows the overall quality value and other information about the project. Second, the U-QASAR platform provides a tree structure where are presented all the elements of a quality model used in the quality monitoring program and their current values. Each of the components can also be viewed in their own window. Finally, the platform provides a customizable dashboard view where the user can select which metric widgets they want to follow.

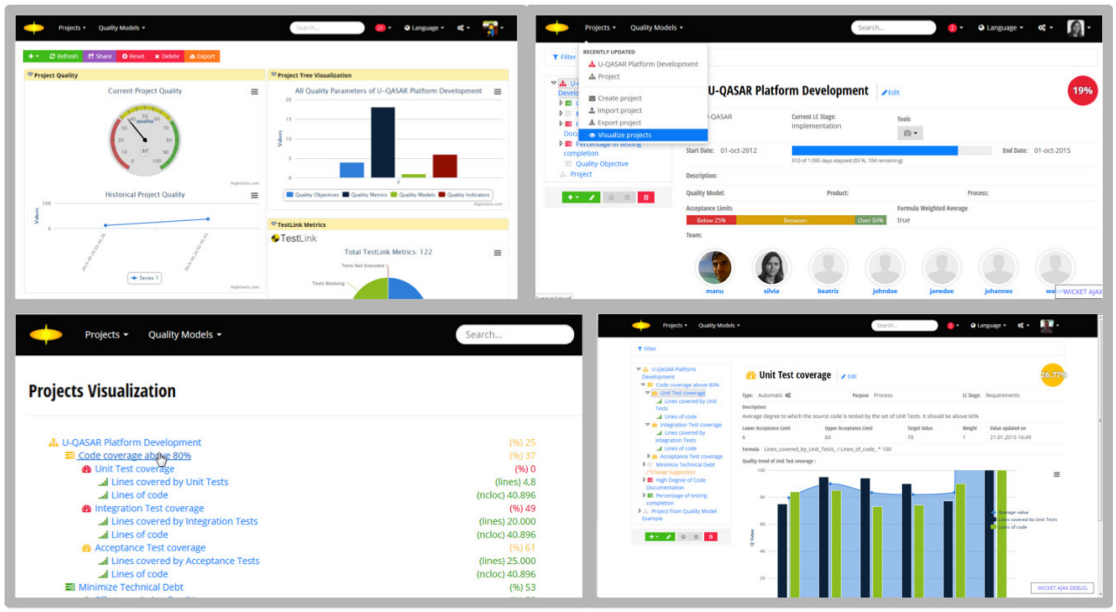

Figure 6. Screen captures from the U-QASAR platform about the monitoring features. Upper-left corner: Customizable dashboard view. Upper-right corner: Project overview. Lower-left corner: Quality model element tree Lower-right corner: Indicator data overview. The images are from the U-QASAR platform design document.

4 Research methodology

This section presents the research methods used in this thesis. Furthermore, the data collection instruments and the data analysis methods are discussed. First, Section 4.1 presents the multiple case study method and its use in this thesis and then Section 4.2 presents the constructive research approach and its use in this thesis.

4.1 Multiple case study

This section describes the multiple case study method and its use in this thesis. First Section 4.1.1 introduces the principles of the method and Section 4.1.2 continues describing how the method was used in this thesis. Section 4.1.3 introduces the background and the settings of each case project and finally Section 4.1.4 describes the data collection and analysis.

4.1.1 Multiple case study as a research approach

In order to explore research questions that solely ask "how" or "why", and not "what" or "how much", Yin (2014) recognizes three main research methods: histories, experiments and case studies. He admits that these methods are overlapping, but emphasizes that histories are mainly for exploring past events with only historical data and in experiments the environment in controllable by the researcher.

According to Yin, the case study is a method for exploring contemporary events, where the "relevant behaviors cannot be manipulated". He states that the strength of case studies over the histories and experiments is the ability to use evidence from different sources, such as interviews and observations. In his book, Yin provides the following two-stage definition for case studies:

1. A case study is an empirical inquiry that
    - investigates a contemporary phenomenon in depth and within its real-life context, especially when
    - the boundaries between the phenomenon and the context are not clearly evident.
2. The case study inquiry
    - copes with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result
    - relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result
    - benefits from the prior development of theoretical propositions to guide data collection and analysis.

4.1.2 The use of the multiple case study method in this thesis

A multiple case study was conducted to explore the real-context use of the U-QASAR methodology and the deployment of a software quality monitoring program and quality monitoring with the U-QASAR platform. The motivation was to gain understanding about the usefulness and feasibility of the U-QASAR methodology in defining a quality

model and to identify the advantages and challenges of using a quality-monitoring tool that integrates data from multiple software measurement tools.

Based on the definition provided in Section 4.1.1, the multiple case study method was used in this thesis to answer the research questions. As the use U-QASAR methodology and platform were studied in a real context, the events could obviously not be fully controllable. Furthermore, interviews or questionnaires were used to explore the experiences of the participants in the study. Thus, the multiple case study method presented by Yin is feasible for this study.

4.1.3 Background and settings of the case projects

This section starts with presenting statistics of the project backgrounds. Then, the projects are described in detail one by one. Five software development projects were selected to represent the case projects. The projects were named Project A, Project B, Project C, Project D and Project E. Project B and Project C were from the same company. An overview of the projects can be seen in Table 8. The case projects took different actions using the U-QASAR methodology and platform, because the schedule and processes in the projects were different. Thus, it is reasonable to describe the settings in the projects.

Table 8. Summary of the case projects.

	Project A	**Project B**	**Project C**	**Project D**	**Project E**
Length	3 years	3 years	3 years	3 months	1 year
Pre-defined quality model	Yes	No	No	No	No
Pre-defined quality	Yes	No	No	Yes	Yes
Process type	Waterfall	Waterfall Prototyping	Prototyping	Agile/Lean	Agile/Lean
Stakeholders	End-users Developers Testers Researchers	End-users Developers Financial	End-users Developers Scientists Communications Authorities	Project leader Developers Reviewer Stakeholder Tester	Reviewer Project leader Developers
Client	Internal	External	External	Internal	External
Number of members	10	2	4	4	4
Member experience	Experienced	Quite experienced	Somewhat experienced	Quite experienced	Quite experienced
Members attending workshop	14	2	2	-	-
Members using U-QASAR	-	2	4	3	1
Use of U-QASAR	-	3 months	3 months	3 months	4 months
Frequency of using U-QASAR	-	Once a week	Once a month	Daily	Once a week

In projects A, B and C, the quality model was defined by the U-QASAR methodology in a facilitated workshop, which will be referred to as *the QOS workshop*. For projects D and E, no workshop was driven. In both of the projects, the top-down approach to define quality was combined to the bottom-up approach so that the quality objectives for the quality model were selected from definitions such as the ISO standard, according to what measurement data was available. By this approach there was no need to refine the quality model for the quality monitoring program. In projects B, C, D and E, a quality monitoring program was deployed with the U-QASAR tool. The deployment was followed by a three-month period of monitoring quality.

In the QOS workshops, also other stakeholders than the project members participated in the quality model definition. This resulted in many different roles among the participants, as can be seen in Table 9. Further background information about the participants is presented in Figures 7 and 8. The most of the participants in both of workshops were well experienced in software development, as is presented in Figure 7. However, also a remarkable part of them had less than 3 years of experience.

Also the participants' experience quality assurance varied. Figure 8 presents the project members experience in defining and using quality information before using the U-QASAR methodology. The variation on their expertise is quite high. The most of the participants had little or no prior experience on defining and using quality objectives, indicators and metrics. In Project A, the most of the participants had strong background in quality assurance and in using quality information. In Project B and Project C, the participants were more inexperienced in the field of quality assurance.

Table 9. Participant roles in the workshops. One person may have had several roles.

Role	**Project A**			**Projects B and C**
	Group 1	Group 2	Group 3	
Engineer/ Researcher			1	1
Requirements Engineer	1		1	
Quality assurance	3	3	2	
Project manager	4	1	2	2
Product manager			1	
Software architect		1	2	4
Programmer	3	3	2	6
Tester	2	1		
Sales representative		1	1	
Customer		1	2	

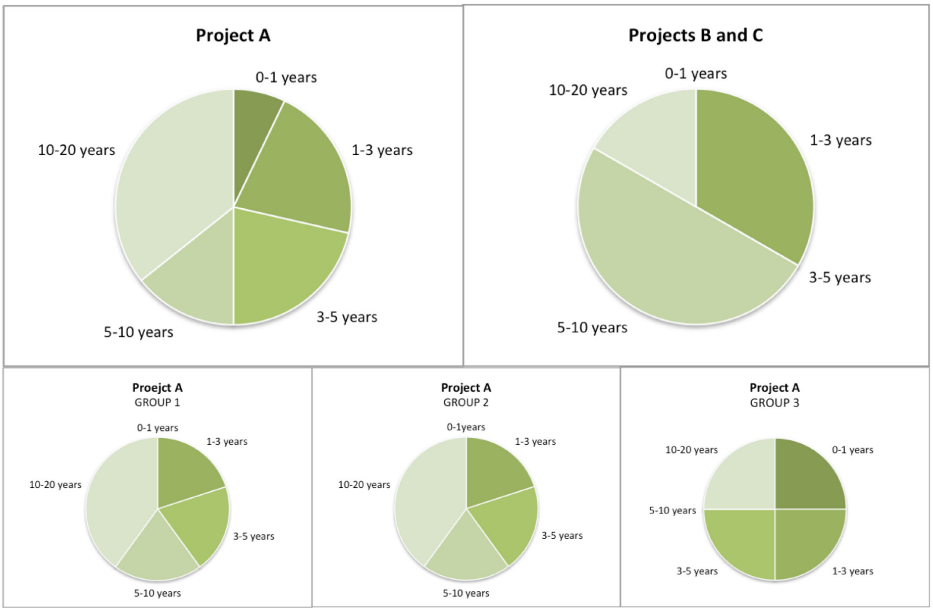

Figure 7. The participants' experience in software development in years.

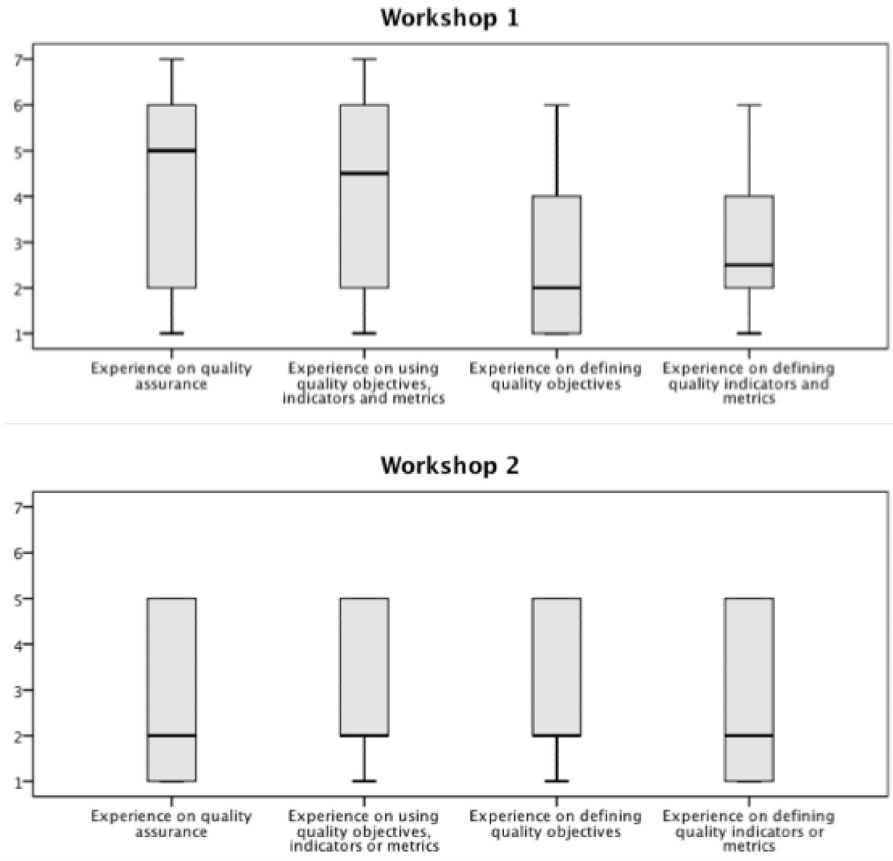

Figure 8. The participants' experience on using and defining quality information.

Project A

Project A was the U-QASAR research project consortium and the development activities concerned the U-QASAR platform. The length of the project was three years and 10 people participated in the software development. Project A used the U-QASAR methodology in a late phase of the project and it did not lead to designing and deploying a quality monitoring program with the U-QASAR platform nor to enhancing an existing one. It rather was a pilot for using the U-QASAR methodology in practice.

The U-QASAR methodology was used to define a quality model for the development project of the U-QASAR platform. A QOS workshop was arranged for altogether 14 people. In this workshop, no training on the concept of software quality was arranged for the participants, but as they were part of the U-QASAR research project consortium, they all were familiar with the concept on some level. Facilitating materials were provided in the workshop, but they were not seen useful.

The participants were divided to three groups to make the procedure smoother and to keep discussions in control. Furthermore, this setting enabled the comparison of the models defined by different groups over each other. Accordingly, three quality models were defined. Discussions were held for each of the groups separately and finally for all the participants together. As the intention of the workshop was to try out the methodology in a real context, the models were not merged and no quality monitoring program was deployed in the project.

The quality models defined for Project A by the three groups are documented in the following ways:

- In Group 1 the defined objectives were categorized by their meaning but the groups were not named. The categories were seen as the quality objectives. The contents of the categories could mainly be seen as the quality indicators.
- In Group 2, a similar categorization was done to the defined items. The categories were named, and these were the quality objectives. The contents of the item categories could mainly be seen as the quality indicators but also some metrics can be identified. Filled elaboration templates provided good indicator-metric structures.
- In Group 3, some categorizing was done, but no clear item groups, objectives, indicators or metrics were named. With the elaboration template two good objective-indicator-metric structures were achieved.

Project B and Project C

Project B and Project C were small subprojects of larger international projects, for which they were developing back-end computing modules. Project B had two members who both were developers and participated in defining a quality model by the QOS method. They were the users of the U-QASAR platform at the monitoring phase. Project C had four members and all of them experimented with the U-QASAR platform. However, only two of them, a developer and a manager, participated in defining a quality model by the QOS method and were the main users of the platform.

For Project B, the goals of using the U-QASAR methodology and platform were to try out a formalized quality management system and to achieve higher software quality. The U-QASAR platform was used in this project for 3 months. As the project members were developers, the intentions for quality monitoring in the project were mainly based on static analysis of the source code. Before starting the monitoring, a reasonable interval for using the platform was seen to be a week, but the platform could be also used as often as it was needed.

Also in Project C, the U-QASAR platform was used in this project for 3 months. The goals of using the U-QASAR methodology and platform in this project were to improve the quality of the software, to recognize missing quality objectives and to reach all of the defined quality objectives. The project members wanted to monitor "all possible measurable indicators". The monitoring interval was suggested to be one month or alternatively the platform could be used after bugs were reported.

A common QOS workshop was arranged for Project B and Project C. In this workshop, altogether 6 stakeholders were present: two members from each project and two software experts from the company. The other member of Project B and the company software experts were also part of the U-QASAR project consortium. The goal of the workshop was to define a quality model for the company, which then could be used in the projects by modifying it accordingly to the projects' needs. The workshop schedule and facilitation materials were enhanced for this workshop based on the feedback from the workshop for Project A and training was arranged for the participants to help them understand the concepts of software quality.

After the workshop, the defined quality model was refined by one of the experts that participated in the workshop and a quality monitoring program was created for both of the projects. As the projects only wanted to try out quality monitoring, only two of the defined quality objectives were included in the programs. The indicators and metrics were selected to be included accordingly to what measurement data was available in the tools used in the projects. As the models were ready, they were inserted in the U-QASAR platform and deployed in the projects.

Project D

Project D was an internal project in a medium-sized international company with more than 150 employees and had four quite experienced participants. The length of the project was three months and the intention was to replace an old server API with a new one. The development process was agile and the stakeholders of the project were the project leader, two developers, a reviewer and a tester. There was some quality documentation available for the project before starting to use the U-QASAR platform: the non-functional requirements, the functional requirements and the metrics.

The users of the U-QASAR platform were the reviewer, the project leader and the developers. The goal of using platform was to ensure the product and project quality and to enable the transparency of the work that was done. The monitoring objects were the statistical source code data, the general progress of the project and the compliance of the project to the process. The monitoring was done on daily basis.

In Project D, three of the project stakeholders defined the quality model by using the available quality documentation as a starting point. Microsoft PowerPoint was used to document the model during the definition. Afterwards, it was enhanced and supplemented by the project members. As the model was ready, it was inserted to the U-QASAR platform and turned into a quality monitoring program by defining the calculations for the quality elements. A training event was arranged for the project members to learn to use the platform and to understand the provided information.

Project E

Project E was a large software project with four experienced participants and the length of about 13 months. The intention of this project was to develop a semantically enhanced search engine to handle about 75 million record documents for an external client. The development process was agile and the stakeholders were the product manager and three developers. This project also had some pre-defined quality documentation: the non-functional requirements were defined and there were some internal code quality objectives. The U-QASAR platform was used in this project for about three months and the main users were the product manager and developers. The intention of using the platform was to have a consolidated view of the code quality issues and to monitor the development and testing. The monitoring interval was set to one week.

In Project E, three managers defined an initial quality model and documented it on paper. They used an ISO/IEC standard as the basis of the definition. By going through each of the quality metrics documented in the standard they selected the most suitable ones for the quality model. The conditions for selection were the suitability for the certain project and the availability of measurement data in the project tools for the certain metric. Afterwards, the project members were allowed to edit the model.

Similarly to Project D, this model did not need refining for the quality monitoring program. When the model was ready, the calculations for the model elements were defined. Then the monitoring program was inserted in the U-QASAR platform and the outputs of the calculations were validated. A training event was arranged for the project members and roles were given to them according to which objectives each of them should monitor and insert manual metrics to.

4.1.4 Data collection and analysis

The research instruments used in the case study were questionnaires, interviews and observations. Table 10 presents the use of these instruments in each of the case projects. As can be seen, the research instruments varied slightly in the projects. This was due to the differences in the schedules and the practices in the projects as well as the project-specific use of the U-QASAR methodology in the projects, as described in Section 4.1.3.

Questionnaires were used when the time was too limited for interviews. Altogether eight different questionnaires were used under five topics: the pre-questionnaire, the post-questionnaire, the project background questionnaire, the follow-up questionnaire and the final questionnaire. All of the questionnaires are documented in Appendix C.

The pre-questionnaire and the project background questionnaire were designed to collect background information about the participants in defining the quality model and about the case projects. The post-questionnaire intended to collect experiences from using the QOS-method, descriptions about different approaches to define a quality model and feedback on the defined quality models. The members of the projects A, B and C filled the project background questionnaire and the pre-questionnaire before the QOS workshops and the post-questionnaire after the workshops. Projects D and E filled all of these questionnaires after designing the quality monitoring program.

The follow-up questionnaire were used to collect experiences and feedback on quality monitoring from projects D and E, as their schedules were too tight for interviews. The final questionnaire was an experience report written by the project members. This approach was used to confirm the researcher's conclusion on the data from the other instruments.

Feedback on the defined indicators and metrics was collected with the quality indicator evaluation form. During the step "Prioritizing quality indicators" in the QOS workshop, the members of Project B and Project C filled in indicator evaluation forms. These forms were a part of the quality objective template presented in Appendix A.

Interviews were used to collect experiences on the quality monitoring and using the U-QASAR platform and they were documented by tape recording. The interview questions are presented in Appendix D and Appendix E. The closing discussion in the QOS workshop for Project A can also be seen as an interview. Also the discussion was tape recorded for further analysis.

Table 10. Research instruments used for each case project.

Instrument	Project A	Project B	Project C	Project D	Project E
Pre-questionnaire	x	x	x	x	x
Post-questionnaire	x	x	x	x	x
Project background questionnaire	x	x	x	x	x
Follow-up questionnaire				x	x
Final questionnaire		x	x	x	x
Follow-up interview 1		x	x		
Follow-up interview 2		x	x		
Follow-up interview 3		x			
Final interview				x	x
Workshop discussions	x	x	x		
Indicator evaluation form		x	x		

Project B and Project C shared their experiences on quality monitoring and using the U-QASAR platform in the follow-up interviews. Furthermore, final interviews were arranged with stakeholders of projects D and E after the monitoring phase. As the project members were not available for the interviews, the interviewed stakeholders discussed with them about their experiences and later shared the learned knowledge in the interviews on the behalf of the project members.

The data analysis was conducted for each case separately and for all cases together as cross case analysis. Both qualitative and quantitative data was collected, which required different data analysis methods. The quantitative data consisted of the scaling question results from the questionnaires and was presented as frequency distributions by boxplot charts. The scaling questions had some variation for the cases due to the different settings and thus the responses to similar objectives were merged to have a larger set of responses. To be able to merge the questions and compare the data from different scaling questions, the scales were kept similar in all of them. The scale was always from 1 to 7 and only the end values had textual descriptions. The lower values of the scale represented negative opinions and the higher values positive opinions.

The qualitative data constructed the majority of the results and two techniques were used to organize and analyze it. First, mind-maps were used to organize the outcome of the QOS workshops. Second, thematic analysis was used to analyze the interview and discussion transcripts and the responses of the open-ended questions in the questionnaires. Braun and Clarke (2006) present thematic analysis as the foundational method for qualitative research. They describe it as a straightforward “method for identifying, analyzing and reporting patterns (themes) within data”. They provide the following six-step guide for conducting thematic analysis:

1. Familiarize yourself with your data
2. Generate initial codes
3. Search for themes
4. Review themes
5. Define and naming themes
6. Produce the report

In this work the thematic analysis was conducted by two approaches. First, the follow-up interview transcripts and open-ended follow-up questionnaire responses were analyzed with the ATLAS.ti tool. Codes were created while reading the materials and finally a network map was created of the identified codes and organized to themes. This resulted in gaining understanding mainly about the U-QASAR platform tool.

The second approach to do thematic analysis was conducted with Microsoft Excel spreadsheets. In this approach, it was first decided what was wanted to be understood from the data, e.g. the positive and negative feedback on the platform or the usefulness of quality monitoring with the platform. Appropriate quotes from the material were collected in a spreadsheet. Then the quotes were given codes and finally the codes were collected in a table. The amount of statements for a certain code was documented, as well as the project bringing it up. The resulting categories of codes were given more abstract names, such as "Positive feedback on the QOS method".

4.2 Constructive research

The deployment of a quality monitoring program with the U-QASAR platform may require implementing a system integration adapter. This section presents the constructive research method that was used to understand the process and effort to implement a data integration adapter for the U-QASAR platform. First, Section 4.2.1 presents the fundamental ideas of constructive research and then 4.2.2 describes how it was used in this thesis.

4.2.1 Constructive research as a method

The constructive research method tries to solve a problem by creating something new and it is a common method in technical sciences (Kasanen et al., 1993). According to Kasanen et al., this method consists of the following five phases:

1. Finding a practical problem with research potential
2. Obtaining a comprehensive understanding of the topic
3. Innovating a solution idea
4. Showing the theoretical connections and research contribution of the solution
5. Examining the scope of applicability of the solution

The basic elements of the constructive approach are illustrated in Figure 9, which is obtained from Kasanen et al. (1993). Besides the practical part, the approach also requires a theoretical part. The practical relevance is achieved by the first step of the method and the practical functioning is achieved from the implemented solution. The theoretical contribution comes from for example documenting the construction process and the theory connection can be achieved if the results are compared to existing documentation or research.

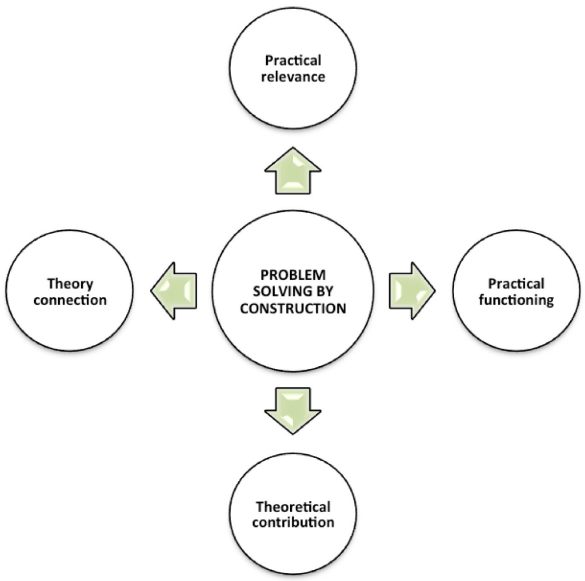

Figure 9. Elements of constructive research (Kasanen et al., 1993).

4.2.2 The use of the constructive research method in this thesis

The first step of the method, **finding a practical problem with research potential**, was achieved by the task setting: the constructive research method was chosen to be used to understand the workload of developing the data integration in the deployment of a quality monitoring program with the U-QASAR platform. Formally, the practical problem was described as following: "*Lack of knowledge on the workload, requirements and challenges in developing a data integration adapter for the U-QASAR platform, in order to deploy a quality monitoring program*".

Before this research was conducted, some data integration adapters for the U-QASAR platform had already been developed. Thus, the second phase of method, **obtaining the understanding of the topic**, could be carried out by first interviewing other adapter developers and then examining the existing adapters. Four developers were interviewed and they were asked about their background, the implementation process and the problems and restrictions that they identified. The interview questions are listed in Appendix F. The interviews were tape recorded for further analysis. Thematic analysis (described in 4.1.4) was used to go through the interview data. The ATLAS.ti tool and Microsoft Excel sheets were used for the thematic analysis.

Innovating a solution idea was the third step of the method and it resulted in developing a data integration adapter between the U-QASAR platform and the Jenkins platform, which is a commonly used continuous integration (CI) tool (Smart, 2011). It offers "Building/testing software projects continuously" and "Monitoring executions of externally-run jobs" and can be extended with a numerous amount of plugins, for example to produce code metric data. The main reasons to implement the adapter for the Jenkins platform were that it is an open source tool, it provides information that can be used for monitoring quality and it has an extensive *application programming interface* (API) for data extraction.

The fourth step, **theoretical connection and research connection of the solution**, was covered by writing an implementation diary during the development activities. The development process, the lessons learned and the challenges identified were documented and later compared to the results of the developer interviews. The gained understanding was then communicated to other researchers in an assessment report on the U-QASAR methodology and platform.

Finally, the constructive research approach suggests **examining the scope of applicability of the solution**. The findings in this research are limited to the other U-QASAR adapter implementations. Other adapters have been implemented for commonly used software development tools and thus the findings can be generalized for the U-QASAR adapters for such tools.

5 Results from the case study

This section presents the results of the case study. Experiences and outcomes from using the U-QASAR methodology and platform are presented. First, Section 5.1 presents the results concerning the usefulness of the U-QASAR methodology and then Section 5.2 presents the outcome of using the methodology. Finally, Section 5.3 presents the feedback on using the U-QASAR platform to deploy a quality monitoring program and to monitor software quality.

5.1 The usefulness of the U-QASAR methodology

This section presents evaluation and feedback on using the U-QASAR methodology. First, Sections 5.1.1 and 5.1.2 present the evaluation of the cost-efficiency and feasibility of the QOS method and the workshop approach for defining a quality model for a software development project. Then Section 5.1.3 presents open-ended feedback on the QOS method. All the results in this section are from Project A, Project B and Project C.

5.1.1 Cost-efficiency of the QOS method

Cost-efficiency was selected to represent the usefulness of the methodology as it was seen that an inefficient methodology consuming a high amount of resources would not be useful for a company. In this thesis, only the cost-efficiency of the QOS method was studied.

The cost-efficiency of the QOS method was defined by the following factors: 1) the effectiveness of the method in making the participants to contribute in defining the quality model, 2) the easiness of sharing information while defining a quality model, 3) the participants view on the cost-efficiency of the QOS method for defining a quality model and 4) the easiness of using the QOS method. The evaluation factors were selected for the following reasons:

- It was seen that the effectiveness of the method in making the practitioners to contribute and easiness of communication make the workshop more effective.
- Easiness of using the methodology helps in adapting it into use and also saves time for achieving the actual goal of the QOS method, a definition of software quality for the institution.

The evaluation results concerning the cost-efficiency of the QOS method are presented in Figure 10. The median value of the evaluation of the effectiveness of the workshop to make the practitioners to contribute varied between five and six, which refers to a high-to-moderate activation rate of individuals at each case.

The most of the practitioners found it easy to share knowledge in the QOS workshop. Therefore it is reasonable to conclude that the QOS method provides the space and means for communication. However, this can also indicate that the amount of participants in the workshops was suitable for such setting: all of the groups using the methodology were small in size, six persons or less in each. For Project A this was implemented by dividing the participants into three subgroups.

The practitioners' evaluations on the cost-efficiency of the QOS method seem to deviate more than the evaluation in other viewpoints. The most of the feedback is positive and between 5 and 6, referring to good perceived cost-efficiency. However, in Project A some of the members rated it to only 3 or 4.

The easiness of using the QOS method was evaluated high; the most of the practitioners found it to be easy to use. This was the case especially for Project B and Project C, for which the structure and materials of the QOS workshop had been enhanced on the basis of the feedback from the piloting Project A.

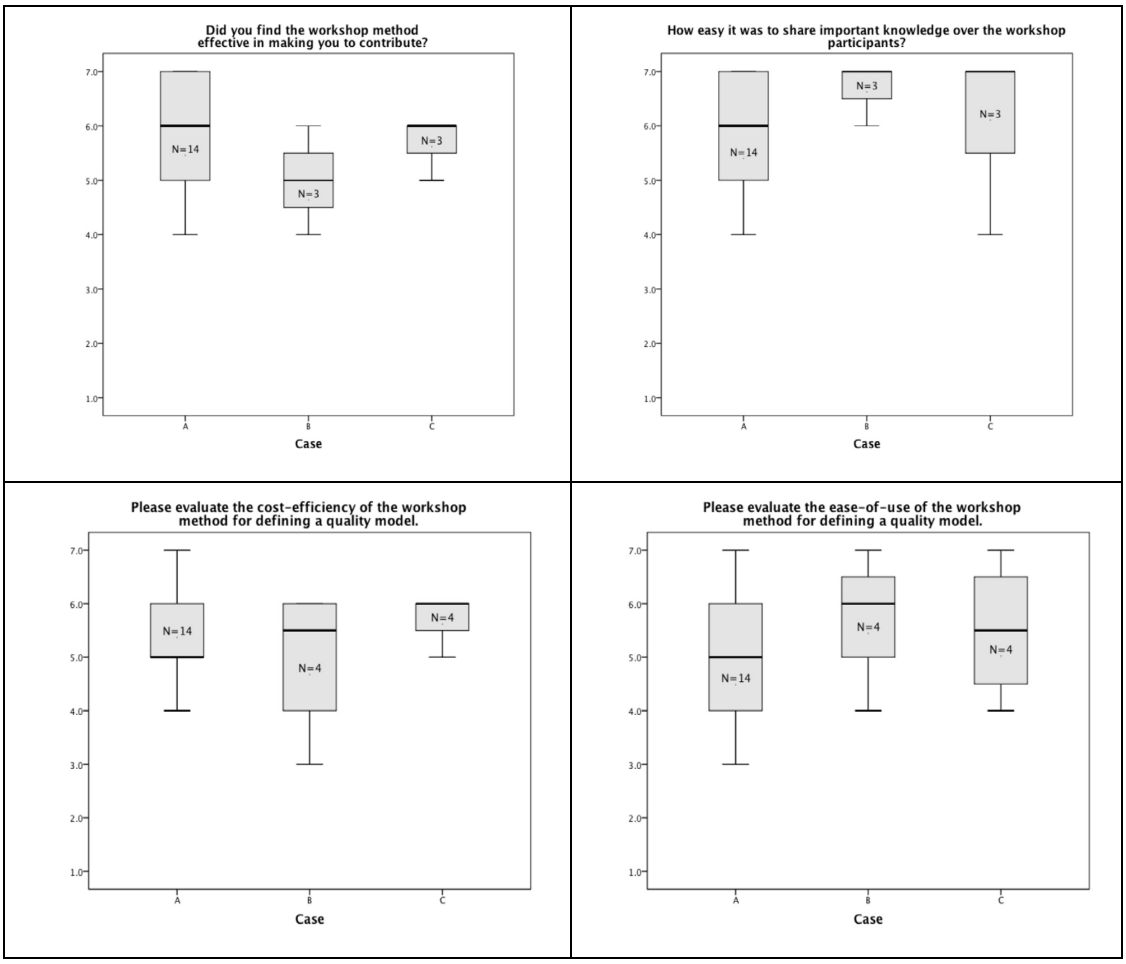

Figure 10. Evaluation on the cost-efficiency of the QOS method.

5.1.2 Feasibility of the QOS method

The feasibility of the QOS method for defining a quality model was also defined by four factors: 1) the general usefulness of the QOS method, 2) the usefulness of the QOS method for the case projects, 3) the feasibility of the QOS method for software projects and 4) the reusability of the QOS method. These factors were selected for the following reasons:

- It was seen that if the practitioners find the method useful for them, the adaptation and use of it is easier, which makes the method feasible for defining quality from the user experience angle.
- It was seen that if the practitioners see that the method is reusable, it means that they find it feasible.

The evaluation results concerning the feasibility of the QOS method are presented in Figure 11. The practitioners evaluated the QOS method to be generally useful for software projects. The practitioners of Project B and Project C found that using the method was useful for their projects. The practitioners in Project B and Project C were asked to evaluate, whether it would be feasible to use the QOS method for every software project. None of them perceived it as a very good idea, because they saw that it might cause a significant overhead for small or short projects.

The reusability of the QOS method was seen to be good; the practitioners were mainly willing to use it in their other software projects. This was expectable because as the members perceived the QOS method to be useful, it supports their willingness to use it again. However, only few of the practitioners found that they would like to use the method "very much" in their other projects, which could mean that there is still something to be improved in the method or that not all the projects are seen to need to use it.

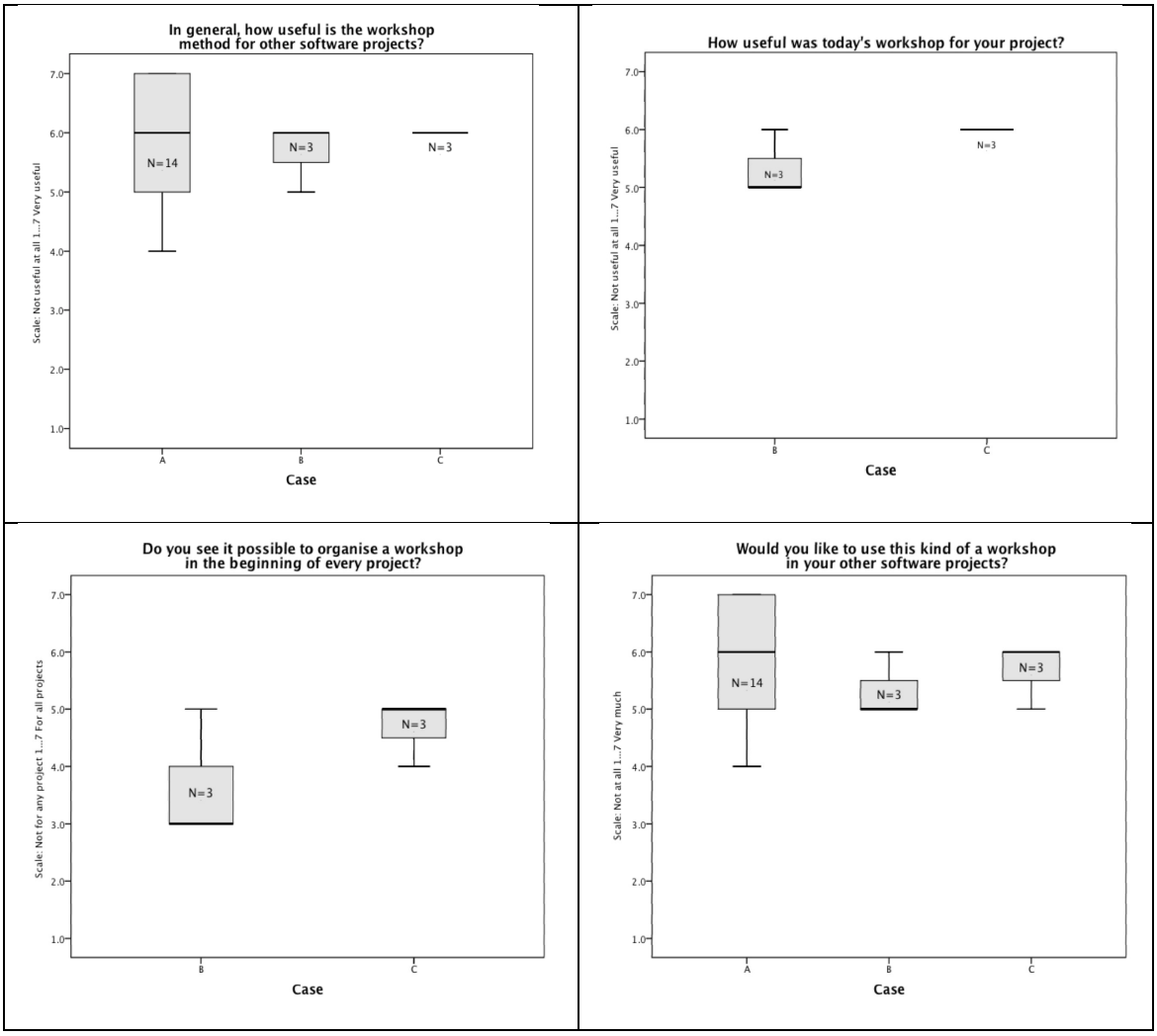

Figure 11. Evaluation of the feasibility of the QOS method workshop for defining a quality model for a software development project.

5.1.3 Feedback on the QOS method

The open-ended feedback on the QOS method is presented in Table 11 and Table 12. Practitioner quotations behind each of the identified concepts are provided in Table 24 in Appendix A. It can be seen that the most mentioned positive terms were *collaboration* and *alignment of the understanding* and the most mentioned negative terms were *the lack of time* and *the lack of guidance*.

It is notable that the QOS method got only positive feedback. The negative feedback concerned the facilitative aspects and the scope of the events where the method was used. Also, the method was enhanced for the workshop for Project B and Project C on the basis of the feedback from Project A. The effect can be seen in the responses: the only negative feedback from Project B and Project C is concerned with the timing of the workshop.

Table 13 presents the lessons learned about quality model definition with the QOS method, identified from the end discussions with the practitioners in Project A. The lessons can be categorized to those concerning the actual definition process and those concerning the people who should do it. To summarize the content of the table, the elaboration of the defined quality objectives is seen to be a heavy process that requires focusing. Despite that it is seen that different stakeholders should participate in defining the quality model, it was suggested that only few core stakeholders should define the initial model. These stakeholders could be for example a technical expert, a product owner and quality assurance expert.

In order to gain a rough understanding of the QOS method's feasibility for software projects, the members of projects B, C, D and E were asked whether a quality model should be defined in the beginning of every project. It was seen as an overstatement to organize a workshop for every project and the following reasoning was presented (number of appearance in brackets):

- A workshop may not fit project's organization. (1)
- There might be too many people in the organization. (1)
- Lack of motivation to work on quality. (1)
- Conducting a workshop is too time consuming. (2)
- Conducting a workshop could be an overhead for small projects. (1)
- A workshop should not be arranged in the beginning of the project but rather as an iterative process during the project. (1)
- Designing a quality monitoring program should be based on an existing quality model. (2)
- Highly meaningful to have a model for each project. (1)

Table 11. Positive feedback on the QOS method.

AMOUNT OF POSITIVE FEEDBACK STATEMENTS ON THE QOS METHOD		
The concept of U-QASAR methodology	**Project A**	**Project B, Project C**
Collaborative approach	5	3
Alignment stakeholders' understanding of quality	3	0
Communication	1	1
Multi-stakeholder approach	2	0
Concept: the method generally	1	0
Concept: focusing the model by voting	1	0
Concept: openness of the method	1	0
Concept: elaboration by iterations	0	1
Facilitation	**Project A**	**Project B, Project C**
Well organized	3	1
Not too many participants	1	0
Content	**Project A**	**Project B, Project C**
Elaborating the results	3	0
Organizing objectives	1	0
Setting context by pre-questionnaire	1	0

Table 12. Negative feedback on the QOS method

AMOUNT OF NEGATIVE FEEDBACK STATEMENTS ON THE QOS METHOD		
Facilitation	**Project A**	**Project B, Project C**
Lack of time	5	3
Lack of guidance: examples of QO/QI/QM	4	0
Lack of guidance: training on terminology	3	0
Poor structure of guidelines	1	0
Content and scope	**Project A**	**Project B, Project C**
Scope: too wide	1	0
Scope: too low-level	2	0
Documenting	1	0

Table 13. Lessons learned from the pilot QOS workshop for Project A.

LESSONS LEARNED	
Defining the quality model	**Project A**
It is difficult to decide on the limits for good and bad	1
The different parts should have different weights on the results	1
Complex metrics need time to be elaborated	1
Participants in defining a quality model	**Project A**
Should be initially done by few people with good technical knowledge	3
All stakeholders should have an influence on the model	2
Customer should not participate	2

5.2 The outcome of the U-QASAR methodology

This section presents the outcome from using the U-QASAR methodology. First, Section 5.2.1 presents the quality models defined for Project A, Project B and Project C by using the QOS method and then Section 5.2.2 presents the quality monitoring programs deployed in projects B, C, D and E.

5.2.1 Defined quality models

The quality models defined for Project A are presented in Figures 30, 31 and 32 in Appendix J. As can be seen in the figures, the contents of the models vary, but they still provide valuable information about quality elements that are seen important for the projects. Furthermore, not many indicators and metrics were defined by quality objective elaboration.

The model defined by Group 2 had the best structure. As clear objectives were achieved by naming the defined item categories, the indicators and metrics defined by the quality objective elaboration are logical. The quality model defined by Group 3 was not as organized as in Group 1 and Group 2 and the defined quality model included fewer elements. Only two quality objective templates were filled which explains the low amount of indicators and metrics.

The common quality model defined for Project B and Project C is relatively extensive. In the definition phase, the categorization of the defined items was done in a similar manner as in Group 1 and Group 2 in Project A and the categories were named to present the quality objectives. The participants decided on five quality objectives to be further elaborated and the quality objective templates were filled for them.

The effect of the described enhancement of training and facilitating materials for Project B and Project C can be seen in the resulting model: it has a good structure with three layers of abstraction and a high number of indicators and metrics have been defined. Furthermore, the documentation of the results was done properly during the workshop, which might also have had an effect on the well-organized outcome.

5.2.2 Evaluation on the quality models

The evaluated aspects on the outcome of using the QOS model were mainly seen in a positive light, which indicates that the participants in the QOS workshops generally found the models good. This is aligned with the responses about the usefulness of the quality models defined by the QOS method, which indicated that the participants were satisfied with the QOS method outcome.

Figure 12 presents the aggregated feedback on the quality models defined by Project A, Project B and Project C. The project members evaluated the defined quality models as useful and covering the project quality well. The only minor negative feedback concerned the completeness of the output as some practitioners found that the main part of the outcome was not ready to be used. The practitioners in Project B and Project C also found that the most of the elements in the defined quality model would require a data integration tool such as the U-QASAR platform to be implemented.

Figure 13 presents the practitioners' feedback on the defined models case by case. The most of the respondents evaluated the quality model defined for Project A to be either useful or very useful. The members of Project B and Project C were asked about the feasibility of the defined quality model to be used. The most of them found the models to be rather covering, but also they all found that the model was not complete yet. The practitioners were also asked to evaluate the need of a data collection tool for collecting the measurement data for the defined models. The responses reveal that the most of the metrics would need to be collected by a data integration tool. This enhances the idea of automatic data collection that is enabled by the U-QASAR platform.

All of the projects A, B and C were asked to evaluate the coverage of the defined quality model. Most of the members found the result of their work to be covering or well covering. The most variation was among the practitioners in the Project A, which might have been caused by the highest number of practitioners and having them split into subgroups. Interesting is also that none of the project members found the output very well covering. This might tell that they feel that there cannot be a perfect model or that the model can change during the project. Furthermore, it might reflect the members' understanding on the evaluation scale; some people give the best rating easier than others.

Project B and Project C evaluated the accuracy and feasibility of the indicators and their metrics defined for the quality objectives. The accuracy was described as how easy the indicator was to implement and the feasibility by how descriptive it was for the objective. Figure 14 presents the rather scattered evaluation results. There are indicators and metrics that are accurate and feasible, accurate but unfeasible or feasible but inaccurate. The most of the indicators were seen somewhat infeasible for the projects, which would mean that the model is not very useful for the project.

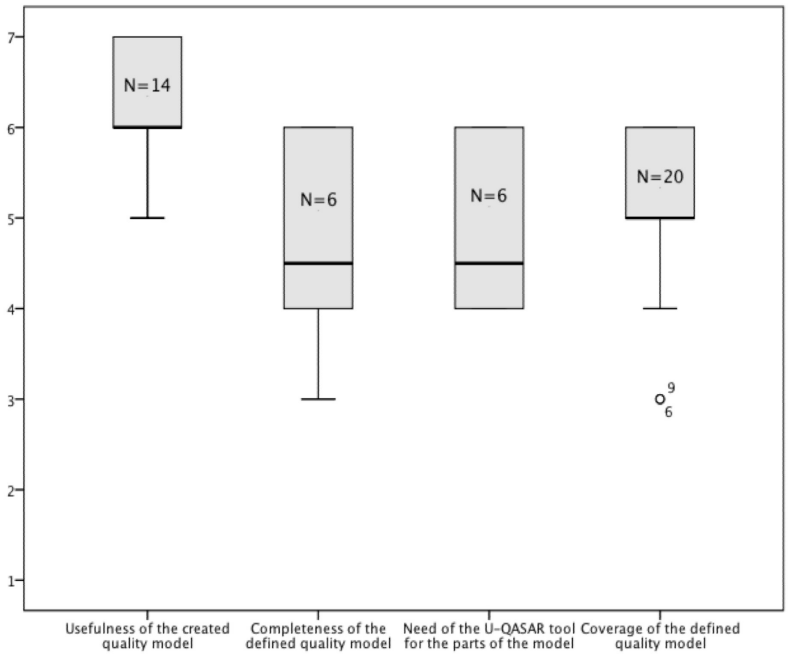

Figure 12. Aggregated evaluation results on the quality models defined with the QOS method.

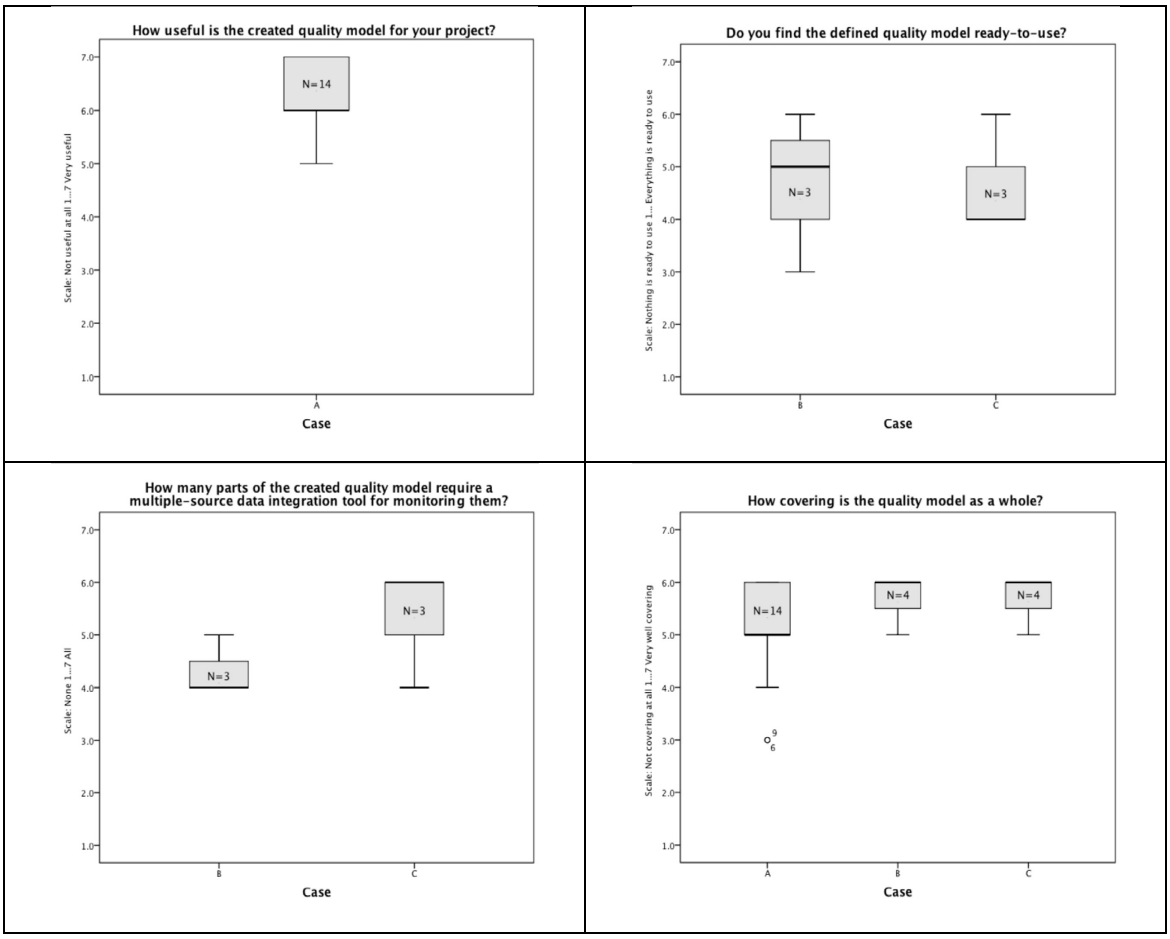

Figure 13. Feedback on the defined quality models case by case.

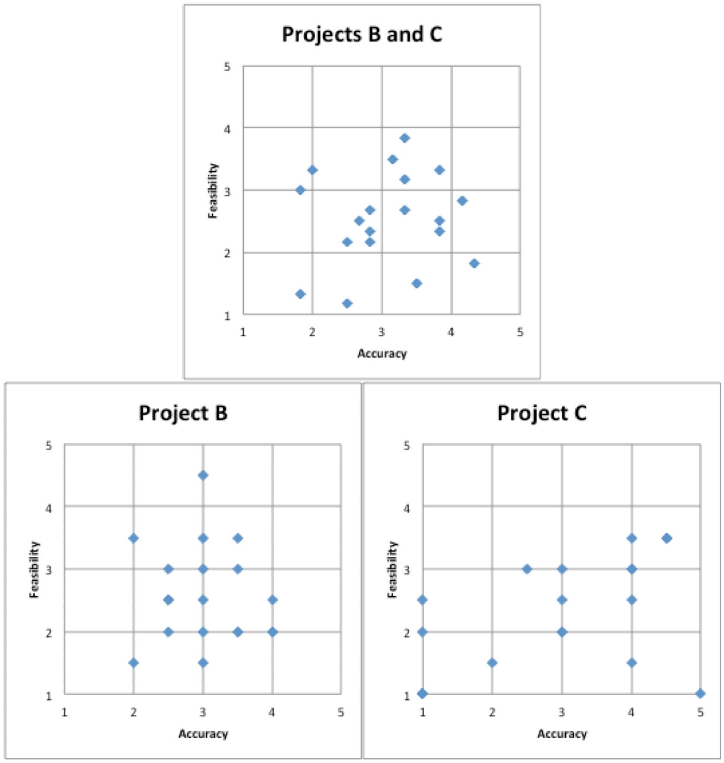

Figure 14. Evaluation on the accuracy and feasibility of the defined quality indicators by the members of Project B and Project C. Each of the points is an average value of the collected evaluations.

5.3 Feedback on the U-QASAR platform

A quality monitoring program was deployed and used with the U-QASAR platform in projects B, C, D and E. This section presents the open-ended feedback on using the U-QASAR platform. First, Sections 5.3.1 and 5.3.2 present the feedback on deploying a quality monitoring program and monitoring quality with the U-QASAR platform. Then Section 5.3.3 presents feedback on the technical features of the platform.

5.3.1 Deployment of a quality monitoring program

The quality monitoring programs for projects B, C, D and E deployed with the U-QASAR platform are presented in Figures 34-37 in Appendix J. For Project B and Project C it was decided to use only two objectives in the quality monitoring program. Despite this, it can be seen from the figures that the implemented programs included less quality objectives, indicators and metrics than the related quality models defined using the QOS method. This means that all the defined quality objectives could not be deployed to the U-QASAR platform.

Table 25 in Appendix J illustrates the usability of the defined quality models well. As described in Section 2.2.2, it is a good practice to start small with the quality model, and thus only the functionality and maintainability of the defined objectives were selected to be measured and monitored. Only five out of the identified 40 metrics were included in the quality models of the deployed monitoring program, meaning that 87,5 % of the defined metrics was ignored. Also, 75% of the defined quality indicators were left out. The main reason for this was that the data for the metrics was not available in the measurement tools that the projects used and manual metrics wanted to be avoided.

Deploying a quality monitoring program means in this context inserting the designed program to the U-QASAR platform, possibly implementing missing data integration adapters for the systems used in the project, introducing the tool to the project team and training the project team to use it. The experiences from the deployment process were collected in the final interviews with projects D and E.

In both Project D and Project E the deployment of the quality monitoring program was seen to be very easy and cost-effective. However, neither of the projects implemented a new data adapter for U-QASAR to support an existing platform already in use in the project or the company. The effort of implementing such adapter is discussed in Section 6.2.

Project D reported that after defining the quality model and designing the monitoring program, the deployment took about 4,5 hours. It included setting up the U-QASAR platform on a virtual machine, inputting the quality monitoring program and conducting a training event for the project members to introduce the platform's features to them. One hour was used for the training and it was seen to be a sufficient time for the project members to learn to use the platform.

Project E recognized a challenge in the deployment process. It was noticed that the semantic alignment of the metrics in the monitoring program is very important in order to understand the output of the model and to maintain the reusability of the metrics, as

one concept or word can have a different meaning among the practitioners, especially over team boundaries. In Table 14, the success criteria table presented in Section 2.2.2 is extended with the column “Project”. All of the success criteria were not in the scope of this study and in this table they are marked grey. A marking of "o" or "x" was given to the project based on if the criterion was used ("x") or if it was not used ("o"). In the following is explained the criteria that was not used in the projects.

The definition of the quality model used in Project B and Project C is discussed previously and as the model was quite reduced in the deployed monitoring programs, in this table it has been concluded that the model was not well defined. Furthermore, Project B and Project C reported that they did not always understand the information provided by the platform and neither could they efficiently use the monitoring program because of the phase of the projects.

Project D and Project E used an incremental approach to define the quality model. None of the projects used internal metric gurus or measurement incentives, as the quality monitoring was still in a prototyping phase. Furthermore, none of the projects was able to implement significant changes in their processes. However, they reported that decision-making based on the provided information would be possible. In Project D and Project E, the manual metric data collection was seen to be quite heavy before a routine for it was achieved.

Table 14. Success factors in deploying a quality monitoring program identified from the literature, extended with the situation in the case projects in this study. In the markings "x" indicates that this criteria was implemented in the project and "o" means that the criteria was not implemented. Grey-colored text means that the criterion was not applicable in the projects during the study.

SUCCESS CRITERIA	PROJECT			
QUALITY MODEL AND COLLECTED DATA	**B**	**C**	**D**	**E**
Well defined and useful quality model	o	o	x	x
Business goal alignment				
Not too large quality model	x	x	x	x
Careful data analysis				
Understanding the collected data	o	o	x	x
Efficiency of quality model and tools	o	o	x	x
Integrity of data				
Proper use of the measurement data				
Practitioners full access to their own data	x	x	x	x
Data privacy	x	x	x	x
PEOPLE	**B**	**C**	**D**	**E**
Practitioner commitment	x	x	x	x
Manager commitment	x	x	x	x
Developer involvement	x	x	x	x
Practitioner training	x	x	x	x
Internal metrics champions	o	o	o	o
External metrics gurus	x	x	o	o
PRACTICES	**B**	**C**	**D**	**E**
Incremental approach	o	o	x	x
Constant improvement of the program	o	o	o	o
Reuse of metrics				
Communication & feedback	x	x	x	x
Transparency of measurement process				
Implementing changes based on the data	o	o	o	o
Monitoring the implemented changes				
Measurement incentives	o	o	o	o
DATA COLLECTION	**B**	**C**	**D**	**E**
Well defined data collection	x	x	x	x
Automated data collection	x	x	x	x
Lightweight data collection	x	x	x	x
Data collection with high frequency	x	x	x	x
Data integration from existing tools	x	x	x	x

5.3.2 Quality monitoring

The U-QASAR platform and the quality monitoring programs were successfully used for quality monitoring in projects D and E. In Project B and Project C this was not achieved and instead, only professional perceptions of using the U-QASAR platform for quality monitoring were collected. This was due to that the projects were in a requirements specification phase and the unsuitability of the code statistic focused quality model for the project activities at the moment.

Experiences on the use of the U-QASAR platform and the quality monitoring program in the case projects were collected from 1) the follow-up interviews with Project B and Project C, 2) the follow-up questionnaire filled by projects D and E once during the time they used the tool, 3) the final questionnaire from all of these projects and 4) the final interviews with Project D and Project E. The findings from these are presented in Tables 15 and 16, and practitioner quotations behind each of the presented aspect are presented in Table 26 in Appendix A.

Positive feedback

The positive feedback, i.e. the feedback supporting the usefulness of the use of the U-QASAR platform for quality monitoring is presented in Table 15. Three main features creating the usefulness of the platform could be identified from the feedback: the data integration from measurement tools, the information that the platform provides and the graphical data presentation approaches in the platform.

The data integration approach in the platform was by far the most appreciated feature as it was seen to increase the cost-efficiency of monitoring the development project. It was seen, that it saved time and effort to have all the project-related data in one place instead of having to use multiple tools. This was mentioned by the most of the projects using the platform and it had the most statements in the feedback by different instruments. Furthermore, it was found that in the U-QASAR platform the information was better available and in a more descriptive format than in other systems, which would enable making better and more accurate conclusions about the project state.

The data that the platform provided was seen useful and valuable. Using the platform was seen to create awareness of the state of the project and to provide tangible proof on what happens in the project. It was seen that the data could be used to justify and facilitate the decision-making on the actions to be taken in the project and on resource management. Furthermore, the use of the platform was seen to give certain assurance on that the project was going to the right direction. It was seen that the use of the U-QASAR platform could generate communication between team members and it could be used for example in team meetings.

Finally, the presentation of the information in the U-QASAR platform was seen useful. It was seen that customization helps different stakeholders of the project to better interpret the information. Furthermore it was seen that everyone should have a suitable amount of information to monitor and that a too high amount of different objectives to look at would not be a sustainable solution. It was seen that this could be handled by the customization of the dashboard combined to proper delegation of the objectives to be

monitored. It was also seen that providing information on different levels of abstraction was useful, as different stakeholders need different granularity level of the collected data.

Other positive feedback included "Enabling efficient decision making," "Comparing projects," and "Communication between distributed teams". The use of the U-QASAR platform was explained to make the decision making more efficient. This would then bring benefits to the company using the platform. Also, projects could be compared with each other to create some competition between them, which was seen to be motivating in the sense of daily development work. Finally, it was seen that the communication in distributed development teams could be enhanced by the use of the U-QASAR platform.

Table 15. Feedback on the usefulness of quality monitoring with the U-QASAR platform.

Feedback	Number of statements	Number of sources	Cases
Usefulness of data integration			
Saving time and effort by having all data in one place	12	5	B, C, D
Saving time and effort by providing simplified data	3	3	B, C
Usefulness of the collected quality information			
Creating awareness of the project state	7	5	B, C, D
Providing tangible proof of success	6	4	C, D, E
Justifying & facilitating decision-making	6	4	D, E
Justifying & facilitating development processes	6	3	C, D, E
Justifying & facilitating resource management	4	2	C, E
Providing assurance of right development actions	1	1	C, D
Generate communication	3	2	B, C
Usefulness of different approaches of presenting the data			
Providing adequate use for stakeholders by customization	5	4	B, C
Providing more abstract information	3	2	C
Providing information on different detail-levels	3	2	C, E

System challenges

Challenges with using the U-QASAR platform were also identified, mainly in the follow-up interviews with Project B and Project C. These challenges could be divided in two categories: user-related challenges and challenges in interpreting the quality information provided by the platform (see Table 16).

Despite that the provided information was seen useful, the practitioners in Project B and Project C found that the quality information provided by the U-QASAR platform was partially difficult to interpret. This issue was mentioned many times in the follow-up discussions as well as in the final questionnaire written by the projects. It was seen that understanding the values of the elements in the quality model was difficult. Furthermore, some project members did not fully understand how the information

would make them better in their work and how the relationship between the provided information and the project actions should be interpreted. It was reported that the calculations and data collection were not understood and that more meta-data should be available in the platform to provide explanations on different items. Also some problems were found in the graphical presentation of the project overview, which was reported as an improvement suggestion.

The user-related challenges mainly consisted of the motivational aspects and the problems regarding the phase of the project, but also the burden of the manual input of metrics was reported. As mentioned before, Project B and Project C were in a requirements specification phase and thus could not really benefit from monitoring quality with a code statistics based quality model. In the interviews and the final questionnaires it was reported that keeping up the motivation to monitor the project with the U-QASAR platform was sometimes challenging.

The lack of motivation for using the platform was also observed by the researcher: it seemed that the projects wanted to participate in trying out something new, but as they did not really achieve much with the monitoring, the motivation was seemingly decreased decreased and it looked like that the platform was used only because the practitioners felt that they had to.

It is notable that none of the projects could report making decisions based on the data and information provided by the U-QASAR platform. This means that the monitoring did not really lead to any concrete actions in the project. However, this can be explained by other factors as well, such as the short observation time. As Project D reported, "--*it was some kind of confirmation for them that they are on the right way*--", meaning that the platform was mainly used to confirm that the development was going to the correct direction, which it apparently was all the time.

Table 16. Challenges in quality monitoring with the U-QASAR platform.

Challenge	Number of statements	Number of sources	Cases
Challenges of interpreting the data			
Interpreting the quality objectives/indicators/metrics	7	4	B, C
Interpreting the results provided by the platform	6	3	B, C
Understanding the data collection and calculations	2	2	C
Understanding the graphics of the platform	1	1	B
User challenges			
Motivation	3	3	B
Phase and size of the project	3	2	B, C
Time used for manual metrics	1	1	D
Change resistance & costs	1	1	B

5.3.3 Technical features

Table 17 presents the requirements identified for a software measurement tool in literature and how the U-QASAR platform relates to them. Only a few things are missing from the implementation: self-monitoring features, co-worker monitoring and interoperation with other software tools. Other lacks in the platform features identified by Project E were the lack of an integrated statistical analysis tool such as R and the lack of a decision support system to support the decision making based on the information provided by the platform.

The output of the thematic analysis is presented as a summarizing table in Table 18 and as a network map in Figure 15. The analysis revealed requirements for a quality monitoring platform and for the provided information, problems that can emerge from deploying quality monitoring and motivators for using a quality monitoring platform, i.e. the user needs for the platform. They were identified directly and indirectly from the interview transcripts and open-ended questions in the questionnaires. "Directly" means that the feature was mentioned by a project member and "indirectly" means that the feature was identified by making conclusions on what had been mentioned.

Table 17. How the U-QASAR platform implements software measurement tool requirements in literature.

REQUIREMENT FOR MEASUREMENT TOOLS	THE U-QASAR PLATFORM
Literature review	
Summaries of data	x
Customized views	x
Measures repository	x
Self-monitoring tool	o
(Madeyski and Majchrzak, 2014)	
Workflow visualization and usability	x
Extensibility	x
Export/import support	x
Ready for commercial use	x
Open source	x
Language and technology independent	x
(Brandtner et al., 2014)	
Ability to locate quality hot-spots in source code	x
Dynamic arrangement of information shown in the user interface	x
Providing awareness of the activities of co-workers	o
Ability to discover changes in real time	x
Interactive visualization of a person's role	x
Ability to interoperate with other software engineering tools	o

Table 18 presents the identified user needs and the related platform requirements. Some of the requirements are not yet supported by the platform and should be taken into account in further development. Such requirements are for example *support for a mobile application* and *comprehensive user support*, including definitions of the platform elements and supportive materials to facilitate the decision making based on the platform. The following challenges were seen to apply to the deployment of the quality monitoring platform:

- difficulty of interpreting the provided quality information.
- burden for project teams with few participants.
- difficulties of inputting the manual metric data.
- reluctance of deploying new systems.
- losing communication by using the tool.

The first two challenges are probably the most difficult to correct but the other can most likely be eliminated by proper training and materials and learning during the use of the platform. The proposed burden for small projects might also apply only in the deployment phase of the U-QASAR methodology and platform in an institution, as it is the most burdening phase with many tasks. However, also the manual metrics can be a problem in small projects. However, the platform could be set up in such way that it is lightweight to use for monitoring. The difficulties in interpreting the information provided by the platform can be a severe problem if it means that quality cannot be defined in such way that the definition is not understood or seen useful by the practitioners. This creates base for further research.

Table 18. User needs and the corresponding requirements for a good quality monitoring platform recognized by projects B, C, D and E. Based on Figure 15.

User need	**Platform requirement**
Assistance in quality model definition	Provide quality model construction by cherry picking
Assistance in interpreting quality information	Provide user support inside platform
Saving time	Provide mobile application Provide data integration
Using only the most interesting data	Provide platform customization
Efficient use of the platform	Provide good usability Provide a satisfying design Provide easy data gathering
Support for project communication	Provide summary reports of the projects Provide information sharing
Data safety in the platform	Provide possibility to set access rights
Efficient quality monitoring	Replace other systems in use
Seeing the "big picture" of the project status	Provide data integration
Provide appropriate quality information	Provide information about technical quality Provide quality information for management needs Provide graphical presentation of quality data Provide historical data Provide information about the progress of the project

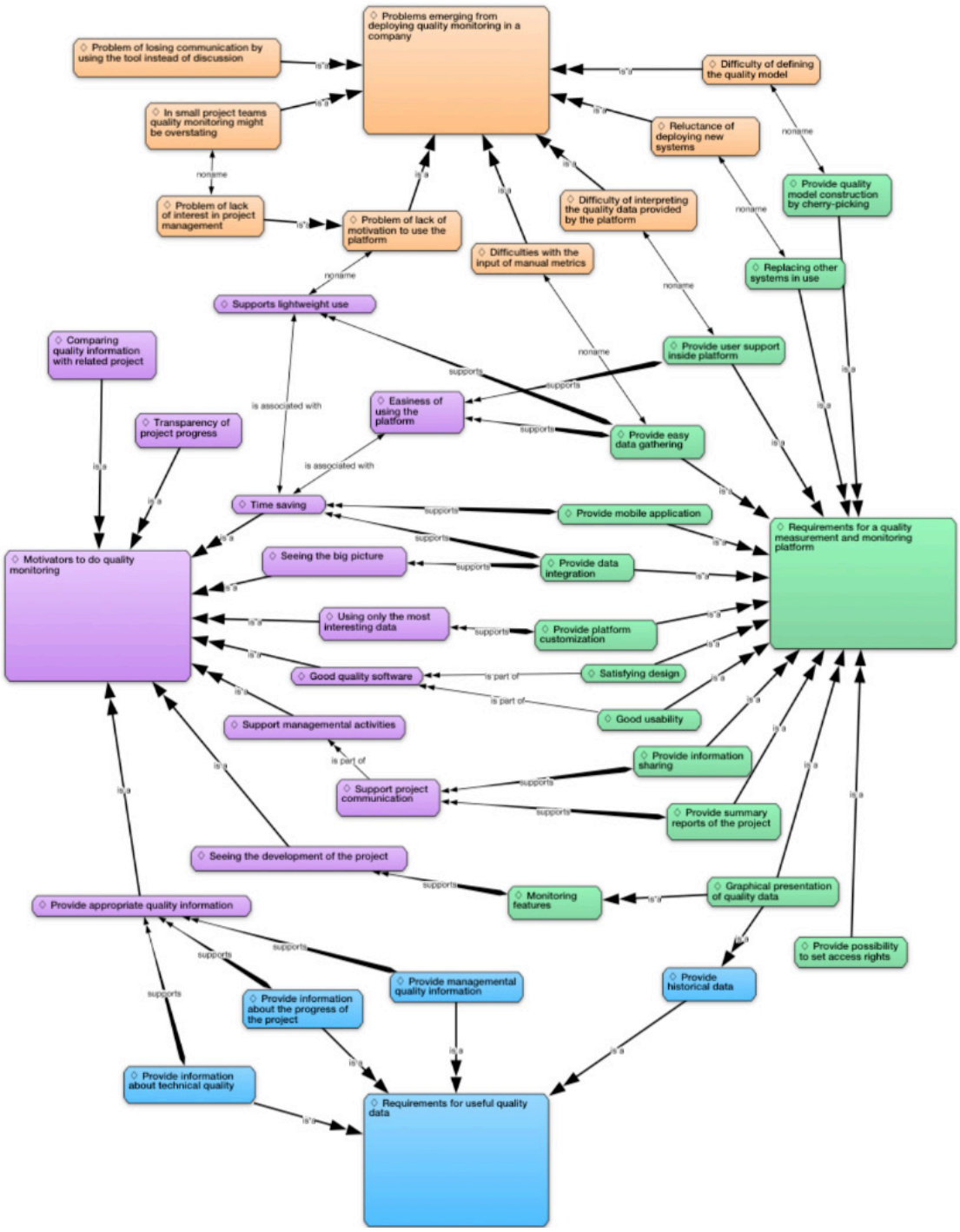

Figure 15. Feedback from projects B, C, D and E on features of the U-QASAR platform.

6 Results from the constructive research

This section presents the results from the constructive research that was conducted to understand the process of developing a data integration adapter for the U-QASAR platform. First, Section 6.1 presents the findings from the interviews with adapter developers and the Section 6.2 continues by presenting the findings from the development of the Jenkins adapter.

6.1 Interviews with adapter developers

To understand the process of implementing a data integration adapter for the U-QASAR platform, the developers of the existing adapters were interviewed. This section presents the findings of the interviews. A summary of the findings can be seen in Table 19. Developer quotations behind the challenges and restrictions in adapter development are presented in Table 27 in Appendix L. The rest of the section describes the findings in more detail.

Table 19. Findings about the implementation process of a data integration adapter for the U-QASAR platform.

Aspect	Findings
Implementation	A mapping between the target system data and the U-QASAR metrics
Time to implement	From the beginning: 3-4 weeks full-time On the basis on another adapter: 2-4 days
Prerequisites	Relevant information from the target system Metrics to be implemented with the data
Matters easing the implementation	Programming knowledge Knowledge on Java language Knowledge about the U-QASAR platform implementation and the target system Experience in using the development environment
Challenges	Privacy of the individual employees working for the institution Varying configuration settings of the target system No API available, connection straight to the database Authentication challenges Negligence on understanding the implementation
Restrictions	The metric data format is restricted Generality of the data that can be retrieved Data processing in U-QASAR platform is constrained The account rights to the target system might be a problem The U-QASAR platform is restricted so that it only accepts refined data that can directly be used in the metrics

The implementation process. The interviewees described the implementation process rather similarly. Generally, it can be concluded that the process consisted of 13 steps as presented in Table 20. Not every developer used all of these steps and it is rather an aggregated view of what the process may include.

The first phase, planning, included reviewing existing adapters when available and creating an initial design of the project, which included definition of the relevant metrics and ideas of the authentication. Thereafter, the interfaces of the data system and the U-QASAR platform were studied, in order to find out how the API is queried and how the data should be converted before transporting it back to U-QASAR.

When familiar with the interfaces, the developers created the methods for the data retrieval by using some existing libraries. For systems that were protected with credentials, also the authentication for the system had to be implemented. Finally, the adapter was tested with unit and integration testing. As the adapters code is not very extensive, the testing was not very extensive.

Table 20: Data adapter development process

Adapter implementation process
1. Review the existing adapters
2. Create a design of the project
3. Study the interfaces (API etc.)
4. Study the structure of a query
5. Get familiar with the data provided
6. Study the data transportation
7. Study the project mapping style from the data system to U-QASAR
8. Study the metric mapping style from the data system to U-QASAR
9. Create the methods for retrieving metrics
10. Study the authentication style
11. Implement the authentication if necessary
12. Do unit testing
13. Do local integration testing

The implementation and timing. In general, the most of the adapter developers agreed that the implementation of a data integration adapter for the U-QASAR platform is a relatively light mapping of data. The first implementation took the most time, about one month full-time work, but the others were finished quicker as they could use the existing adapters as the reference. Two of the implementers told that it had only taken a couple of days to get a new adapter to work. However, the time needed for the implementation heavily depends on the available resources for creating the connection between the systems.

Prerequisites. According to the developers, there were only two main prerequisites for implementing a data integration adapter. First, the system to be integrated should provide some relevant metric data for quality monitoring, i.e. the developer should know what data should be retrieved. Second, the data should be in such form that it could be converted to the "measurement object" form that U-QASAR uses. However, as

these two matters were the real prerequisites to the implementation process, the developers mentioned several things that would be an advantage for the development and that would considerably shorten the development time.

Matters easing the development. The most important matter that could ease the development was that target system would provide a well-documented API for the data extraction, which would eliminate or at least limit the need of making calls directly to the database. Moreover, it was mentioned that existing libraries for the API would ease the adapter development even more, as it would decrease the need for implementing methods for the data extraction. Naturally, also the existing adapter implementations were mentioned to ease the process, as another adapter might only need small modifications to fit the target system.

As the interviewees had a background in software development and all of them had at least some experience in Java programming, they did not have to use time for such matters as setting up the development environment and getting familiar with Java documentation. It was said that in case the environment should have been set up as well, a lot more time would have been used.

Challenges. The interviewees recognized some matters that could cause problems in the implementation process and with the existing adapters. As mentioned before, the existing adapters were used as reference for the implementation. In other words the adapter code was copied and the required modifications were made to it. One of the interviewees made a point, that this kind of way of work might cause negligence to the possible errors that have been made on the reference adapter and it also hinders coming up with new, more effective implementation. In despite of using the existing adapters as a reference, it was stated in the interviews that there is not a general design for the adapters, which can lead to maintenance problems with the code.

An example of negligence towards the adapter code could be the authentication with credentials to the target system, which was also declared to be a possible security issue. It is a platform-related issue, but the adapter developers should be aware of it too. Also the privacy of developers in the project using the U-QASAR platform was seen as a concern. If the quality data retrieved from the target system is too detailed and the problems with the quality traceable to a certain developer, it would be a problem according to one of the interviewees.

A more practical problem in the implementation recognized by the interviewees was that there might not be a proper interface in the target system. This could cause considerable additional work, as the query methods to for the database would have to be implemented from the very beginning instead of using existing libraries. However, this was not seen as a restriction for the implementation, but rather an additional workload. Other practical problems mentioned were configuring the data system and the cases where a data system uses another data system to collect data. The data system configuration was found to be an issue in case the (API) query methods would fail because of inappropriate system configuration.

Restrictions. Some restrictions in the platform and the adapters were identified by the interviewees. One of them was the generality of the platform implementation, which restricts the use of the data available from the measurement tools. For example, the platform is only capable of using project level data. This might not be a problem in small projects with a rather simple project structure, but in larger projects having subprojects with several teams it might cause issues. Another platform-related restriction was found to be the lack of possibility to process and use the metric data inside the platform, for example to make queries based on data retrieved before. However, as these issues are related to the platform implementation, they can possibly be solved by further development of the platform.

6.2 Findings from adapter development

Developing a data integration adapter did not take long after making some preparations on existing materials and setting up the development environment. The following mandatory preparations had to be made:

- the workspace in Eclipse was set up (settings and configuration of Maven)
- the JBoss server and local hosting were studied
- SVN, Git and GitHub repositories were studied
- existing adapter implementations were studied
- the Jenkins platform and its API were studied
- suitable libraries for using the Jenkins API were searched for

It did not take long to develop the adapter as all the preparative actions were taken and the researcher felt comfortable with the configuration. The authentication with the Jenkins platform could be done by the methods implemented in a library. Then the data that the API provided via the library was analyzed and based on that it was decided which metrics would be implemented with the adapter. The development itself took about a day, which included sorting out the authentication and implementing the metric mappings. The following metrics were implemented with the adapter:

- the state of the latest build
- the state of 100 latest builds
- the information of all the projects in the Jenkins instance

Finally, the adapter was integrated to the U-QASAR adapter pool and a U-QASAR specialist tested the functionality. A screen capture of U-QASAR widgets implemented with the Jenkins metric data can be seen in Figure 16. The source code of the main functionality of the implemented adapter is presented in Appendix A.

The main findings from the constructive research are presented in Table 21 in a similar format as the findings from the interviews. The results are aligned with the interview responses. It was understood that the adapter is a data mapping of data from the target system to the U-QASAR platform. This means, that in order to implement an adapter for any system it has to be known what data the system provides that can be used for quality monitoring. In other words, before implementing an adapter it should be known what quality information is relevant for a software project.

The technical requirement for the system was that it would in first place provide an access to the data. The best option for the data access would be an API with suitable libraries. If these are not available, it will make the development process remarkably longer as all the methods have to be implemented first before they can be used. The requirements for an adapter developer are

- programming skills,
- good understanding of software quality and
- good understanding of the institution using the adapter

These requirements apply if no materials and help are provided to the developer. If the work is done in cooperation with e.g. a quality management team, then the experts can provide the quality knowledge and the institutional aspects to the adapter developer.

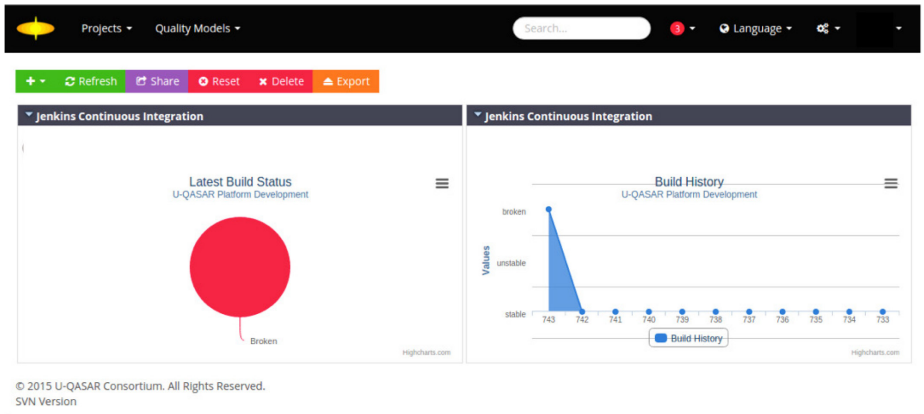

Figure 16. A screen capture from the U-QASAR platform: widgets presenting quality information retrieved from Jenkins.

Table 21. Observations from the constructive research.

Aspect	**Research observations**
Implementation	Mapping from Jenkins data to U-QASAR data objects
Timing	14 days of preparation, 4 days for implementation
Pre-requisites	Knowing what metrics are needed Knowing what data the system to be integrated offers Some programming skills if doing the implementation on an existing adapter Good programming skills if doing the adapter from beginning An interface for retrieving the data from the system to be integrated
Matters easing the implementation	Good programming skills A system that provides ready metrics Extensive API Libraries for using the API or other interface
Challenges	Technical challenges with the environment Deciding what metrics should be implemented
Restrictions	U-QASAR platform only takes in ready metric data and it is not possible just to open the data connection to the system to be integrated.

7 Discussion

In this section the results of this study are discussed. This section begins by answering the research questions in Section 7.1. The Sections 7.2 and 7.3 present the academic and practical implications of this study. Finally, Section 7.4 discusses the validity and reliability aspects of this work.

7.1 Answering the research questions

The best approach to analyze the research problem, "In software development, is it reasonable to utilize a quality monitoring methodology using multi-resource data integration?" is to answer the research questions that were set in Section 1.2.

RQ1. Is the U-QASAR methodology perceived feasible for designing a quality monitoring program?

As the term “feasibility” can have different meanings depending on the context, it is necessary to first define what it means in this research question. It was determined that the methodology would be feasible, if the practitioners found it feasible, it was cost-efficient and could provide a good outcome to base the design of the quality monitoring program. These criteria were evaluated based on the results of 1) the practitioners’ evaluation of the cost-efficiency and the feasibility of the QOS method for defining a quality model, 2) the evaluation of and observations on the quality models defined with the QOS method and 3) the effort and success of designing the quality monitoring program based on the defined quality models.

The results show that QOS method provided by the U-QASAR methodology is feasible for defining software quality in development projects similar to projects A, B and C. This supports the existing literature on the strengths of the GQM-based methods for defining quality. However, there are still challenges in defining the connections between the element layers as well as in determining suitable indicators and metrics for the quality objectives. Good material can be collected for a quality model by using the QOS method, but is not enough to accomplish a full quality model. This leads to that some changes to the structure and facilitation of the QOS method should be considered. These findings are discussed in the following.

Practitioners’ evaluation of the QOS method

The practitioners evaluated the cost-efficiency and the feasibility of the QOS method to be high. The method was seen relatively easy to use and many of the practitioners were willing to use it again. They also perceived it as cost-efficient. However, it was seen to add overhead especially to small projects if it was to be used before every project without reference models, such as a company model. Thus, existing models should be reused after defining them. This result could also imply that the U-QASAR methodology is not yet lightweight enough to support the smallest software projects. Is quality such a complex and heavy concept that defining it cannot be lightweight?

This finding enhances the idea of having modifiable quality models on the institution level to facilitate defining project-specific models, and the possibility to reuse existing quality models instead of every time starting from the beginning. Especially for very short projects with a low number of participants it might not be cost-efficient to define a quality model from the beginning and implement a quality monitoring program based on it. Thus, also the concept of the U-QASAR platform is justified: it supports saving, exporting and importing quality models used in quality monitoring programs, which helps reusing them.

Some practitioners in Project A had differing opinions on the easiness of using the QOS method. This might be caused by the piloting-phase of the methodology and that the facilitative materials and scheduling of the event were not yet as optimized as possible. The practitioners of Project B and Project C received a training on the concept of quality and facilitating materials before the workshop and also evaluated the easiness of using the QOS method higher than the practitioners of Project A. It seems to be important that the practitioners in the workshop to attend a training on quality concepts so that they will be able to use similar terms in communication.

Observations and practitioners' evaluation on the defined quality models

The practitioners in projects A, B and C evaluated the quality models that they had defined to cover the quality elements in their projects well and the practitioners in Project A found that the defined quality model was very useful for their project. Thus, it seems that the practitioners were satisfied with the work they had done. Despite to that, problems can be identified in all of these quality models.

The models created for Project A were quite unstructured and not many links were defined between the elements on different layers of abstraction. In Project B and Project C, the practitioners found that the defined quality models were not entirely ready to use. Furthermore, as they evaluated the descriptiveness of the indicators and metrics and the easiness to implement them, it was realized that many of the indicators actually did not indicate the state of the objective that it was defined for and that many of the metrics did not suite the indicator that they were defined for.

Based on the previous, it can be concluded that it was useful for the projects to define a quality model and they were covering at least on the quality objective level, but the problem was to define descriptive indicators for the objectives and to find objective and implementable metrics for these indicators. Although the facilitation and timing of the workshops may have had an influence on this result, it is likely that the cause is the pure difficulty of defining descriptive quality metrics. This difficulty has been identified in literature as well: Kaner and Bond (2004) concluded that there are "too many simplistic measures" that do not measure what they are supposed to measure. However, if the participants find the defined quality model useful for them, they are more willing to use the method again in other projects and thereby get more experience and simultaneously create more reference material for the definition, such as lessons learned and defined quality models for future workshops. This could possibly lead to more objective results.

Observations on the quality models in the deployed quality monitoring programs

In Section 5.2 it was presented that the most of the elements in the quality models defined for Project B and Project C were not included in the deployed quality monitoring programs. Although it was decided to only use two of the quality objectives in the models, many of the indicators and metrics defined for these objectives were not used. This reveals problems in the U-QASAR methodology and platform.

The cost-efficiency of defining a quality model with the QOS method decreases if the defined quality elements are not feasible or cannot be used. Furthermore, it can cause the practitioners to feel unmotivated if they find that their time and effort are not valued. Finally, the U-QASAR methodology does actually not include much advice on how to efficiently use all the elements in the defined quality model. Despite to this, the practitioners in Project B and Project C seemed satisfied on having defined quality in their projects as they learned something about it.

The problem in deploying the defined elements can also be caused by the U-QASAR platform. If manual metric collection is wanted to be avoided and the automatically collected data does not cover the metrics in the quality model, some quality elements have to be left out of the quality monitoring program. A solution for this could be to further develop the U-QASAR platform in the way that using the manual metrics becomes more cost-efficient.

In projects D and E the practitioners were satisfied with their quality models in the deployed quality monitoring programs. In both of these projects the quality model was defined incrementally so that an initial definition was done by a small group of quality assurance specialists and managers and then the model was edited and refined by the project members. This was also suggested in the discussions in the workshop with Project A. Moreover, the literature suggests the incremental deployment of a quality model. This could provide a base for enhancing the QOS method.

The effort and success in turning a quality model into a quality monitoring program

All of the projects B, C, D and E faced challenges when designing the quality monitoring program. In Project B and Project C, it was the lack of implementable elements in the quality models. Project D has a relatively small model and it is difficult to say whether it really covers all the important aspects of the quality. In Project E, the main difficulty was to come up with feasible data sources for the metrics and to define and validate the formulas between the element layers in the quality model of the monitoring program. As a conclusion, the objectivity of the quality model elements and resolving the connections, i.e. the calculation formulas between them seem to be the main challenges in designing a quality monitoring program.

RQ2. Is the deployment of a quality monitoring program perceived as cost-effective with the U-QASAR platform?

The deployment of a quality monitoring program with the U-QASAR platform without implementing a data integration adapter is easy and lightweight. The additional workload caused by the adapter implementation depends on the target system or systems and the amount of the required adapters. The discussion on the adapters and the recommendations concerning them are presented in the context of RQ4.

Experiences of the deployment of a quality monitoring program with the U-QASAR platform were collected from projects D and E. Based on their feedback and evaluations presented in Section 5.3.1, the U-QASAR platform provides a very easy way to take a well-designed quality monitoring program into a use in a development project. It was seen that inputting the defined quality model into the platform was "extremely easy" and that the training of the practitioners did not take more than a couple of hours. However, it was highlighted that defining the quality and designing the monitoring program was not as easy and straightforward and it required significant efforts to get into the stage of deploying the program.

RQ3. Is a quality monitoring program deployed with the U-QASAR platform perceived as useful for monitoring software quality?

The word “usefulness” can have different meanings and so it is first necessary to define what it means here. A quality monitoring program deployed with the U-QASAR platform is useful if 1) something beneficial for the project can be achieved with it, 2) the practitioners can identify useful features in it and 3) the benefits of it are greater than the challenges or problems.

The results show that a quality monitoring program deployed with the U-QASAR platform is useful, especially for having evidence on management decisions and for more efficient use of the project measurements as they are collected in one place and on different levels of abstraction. The challenges included interpretation problems and the issue of manual metrics. As it is likely that these challenges can be overcome, it can be concluded that the benefits are greater than the challenges.

Benefits of quality monitoring

The practitioners in Project B and Project C could not use their quality monitoring programs to monitor the quality of the processes and the product and no real benefits could be reported. Projects D and E reported that they were not able to make decisions based on the quality monitoring. Thus, the real benefits achieved by quality monitoring cannot be evaluated. However, all the projects hypothesized that they would be able to make decisions based on quality monitoring, if high deviation or undesired changes were detected in the data trends. Furthermore, all of the projects reported that it is beneficial to have all the measurement data in one place and on different levels of abstraction because it saves time and effort in managing the state and direction of the project.

Useful features

The useful features identified in quality monitoring with the U-QASAR platform are presented in Table 15 in Section 5.3.2. The U-QASAR platform in its current format was seen to be useful in both monitoring quality and monitoring the progress of the project. Two most useful features were the data integration, which increased the cost-efficiency of the monitoring activities and the production of historical data of the progress of the project, which enabled tracking changes in the values. In addition the practitioners reported ten different useful features, mostly for project management purposes. Especially justification and facilitation of management activities with the collected data was seen useful.

Challenges

The challenges identified in monitoring quality with the U-QASAR platform are presented in Section 5.3.2 (see Table 16). The most of them was collected from Project B and Project C and are therefore mostly hypothetical. The most significant challenge was seen to be the interpretation and understanding of the data provided by the platform. If the practitioners are not able to understand and interpret the collected data, the most of the benefits of the monitoring actions are lost. However, it is possible that this problem can be solved by providing more training and supportive materials for the practitioners. Furthermore, if the semantics of the quality model elements in different models can be established in a systematic way, it helps to understand different models.

Project D mentioned the problem of the burdening manual metrics and the topic also came up in some discussions with Project B and Project C. This problem will be further discussed with research question 4 but it also adds to this topic. If many of the metrics have to be inserted manually and the practitioners are unwilling to do it, the usefulness of the quality monitoring decreases.

RQ4. What improvement factors can be identified for the U-QASAR platform?

Based on the interviews and the questionnaire responses, the overall user experience of the U-QASAR platform seemed to be good. Improvement factors for the U-QASAR platform were identified both in the case study and in the constructive research.

As the U-QASAR platform was in a pilot phase when this research was conducted, many comments on the features of the platform were provided, as presented in Section 5.3. Five notable improvement suggestions can be identified in the case study: 1) more extensive user support in the platform, 2) a mobile application, 3) notification support, 4) enhanced information sharing between the users of the platform and 5) more advanced data integration between the U-QASAR platform and the measurement systems.

Features 1-4 are relatively easy extensions to be implemented and some of already exist: some user support was added to the platform after the feedback came up and, and the notifications and the mobile application have been initially implemented, but would need further development to become fully working features.

In the constructive research it was found that the data integration adapters used in the U-QASAR platform are mappings of data that may also include some calculation. This can cause differences in adapters developed for the same target system, as the semantic meaning of the metrics can differ, as well as the understanding of what calculations should be included in the adapter. Furthermore, the adapters need to include the knowledge of which specific metrics are being retrieved from the target system. Thus, also the configuration of the target system can significantly affect the adapter development. Consequently, the reuse of the adapters can be difficult in different companies and projects.

The authentication in the current system architecture also creates an issue. As described by one of the interviewees, the username-password authentication method currently used in the adapters is not the safest option and would require changes. This is a critical issue in the acceptance of the platform in companies that have measurement tools running outside their internal network, as data safety is very important.

The data collection practices in the U-QASAR platform need to be enhanced, as it now requires either sufficient measurement tools or manual data inputs. If the project does not use many measurement tools, there might not be nearly enough quality metric data available. This causes a situation where more measurement systems should be taken into use in the project or the data should be collected and inputted manually, which can decrease the willingness of deploying the U-QASAR platform. This weakness of the platform was faced for example in Project A that used Excel to track bugs instead of a specific software.

7.2 Academic implications

This section presents the findings of this thesis that are seen to relate to the literature on software quality and software measurement. This study did not add to the existing knowledge on software quality, but it strengthens many parts of it. The implications on the literature can be drawn from four topics: 1) the literature on software quality, 2) knowledge on quality monitoring, 3) knowledge on defining quality and 4) knowledge on quality models.

The literature on software measurement and software quality is fragmented. This thesis contributed in the literature on software quality monitoring by providing a covering literature review. The process of conducting the literature review revealed that the related literature is wide and rather fragmented. Moreover, the terminology on the software quality research is varying, which makes it difficult to find and interpret the articles.

Despite the extensive amount of research papers, the researcher could not find appropriate literature on the process of defining a quality model and designing a quality monitoring program based on it. The research mainly contributes to creating a software measurement program and even the term “software quality monitoring program” had to be introduced and defined for this thesis.

Quality monitoring should be customized. Garvin's (1984) idea of different views on quality was strengthened as in this study it was found that the idea of customizable monitoring features is feasible for quality monitoring in software projects. As presented in Section 5.1.3, the open-ended feedback on the QOS method highlighted the feedback on the benefits of the collaborative approach in defining a quality model. Furthermore, the alignment of stakeholders' understanding about quality was mentioned to be a positive feature.

It was also found that the need for the granularity of the information is different for the stakeholders. The developers may need very detailed information to identify the root causes of their problems whereas the managers may not even understand the details and require more abstract pieces of information. Also, the developers use the information to correct their own actions, but the managers use it for seeing the big picture of the project.

The practitioners need training to understand quality. From the three issues in deploying a measurement program presented by Briand et al. (1996), this study supported the idea of the "Training the personnel to understand quality measurement" to be important in the deployment. The practitioners in projects D and E were provided with a more extensive, hands-on training than the practitioners in Project B and Project C, which probably caused the better understanding of the provided information in projects D and E. This was deduced from the contents of the negative feedback on the use of the U-QASAR platform; Project B and Project C presented challenges in understanding and using the quality information, whereas projects D and E only saw the manual metrics as a problem. Furthermore, in the QOS workshops it was found that introductory training to software quality helped producing better quality models.

The top-down approach is feasible for defining quality. In literature, the top-down approach has been found feasible for defining software quality (Briand et al., 1996). In this research, the goal-driven approach on defining software quality produced good results in two manners: the defined models were rich and wide and the practitioners were satisfied with the results. Furthermore, the defined metrics were connected to the high-level objectives. Although the most of these connections were later found unfeasible, the idea was understood, as can be seen on the structures of the models defined for Project B and Project C. With further work on the models better connections could be achieved.

Quality models should be defined incrementally. The literature suggests implementing a software measurement program incrementally (Coman et al., 2009; Frederiksen and Iversen, 2003), meaning that first only a few measurements are implemented and the program is increased as time goes on. In this study, it was found that a similar practice should be used for defining a quality model. The success of turning a quality model to a quality monitoring program was better in projects D and E, where quality experts defined an initial quality model and it was then modified by the practitioners. This might have improved the understandability of the model and the resulting information.

Quality information has to be reasonable and useful. Coman (2009) presented that quality information should be useful to make a measurement program succeed. This study strengthens the idea. Project B reported that it could be annoying to use the U-QASAR platform. This can arise from the level of usefulness of the provided quality information in their quality monitoring program: as described, in Project B and Project C the platform was used in a non-development phase, which caused that the provided information was not useful for the practitioners at the moment.

Relationships between quality model elements are difficult to construct. The challenge of defining feasible connection between quality model elements presented in Section 2.1.3 is confirmed in this thesis. As described, in Project B and Project C it was found that the defined connections between the elements in their quality models were not accurate or descriptive. Furthermore, Project E reported that defining the calculation formulas for the quality monitoring program was challenging. Thus, it can be said that determining objective measurements for high-level quality definitions is difficult.

7.3 Practical implications

This section presents the findings of this thesis that are seen to relate to the practices in software quality monitoring. This are from two categories: permanence of quality monitoring tools and data integration.

Open source quality monitoring tools do not last in use. Based on the literature, it seems that the tools developed for modeling and monitoring software quality do not last long in the awareness of the target users. There can be many reasons for this. For example, it could be that the developed tool has not been provided with enough use instructions, software development organizations have not heard of the tool, the tool has been licensed so that it does not fit the organization or in-house tools have been developed. Furthermore, it always requires effort to explore whether a tool is suitable for an organization's needs or not. Hence, the organization might rather consider buying it as a service or developing an own tool.

The means for preserving the U-QASAR platform in active use of software development organizations should be studied and applied. The good usability and ease of use of it could encourage companies to take it into use, but branding and marketing are required to achieve awareness of a new tool. Also continuous further development should be done to make the tool better and to keep up with the newest technology and innovations. This could be achieved by an open source license.

Software measurement data integration increases the cost-efficiency of project management. In the case study it was found that it is a good idea to collect software measurement data into one central system and present it on different levels as it makes the information to be more easily available. Furthermore, collecting metrics manually was found to be decreasing the cost-efficiency of the quality monitoring. Adding to this, it was reported that the data provided by the U-QASAR platform guides the project management actions. This implies, that the software measurement data should be collected and presented centrally.

The current U-QASAR data integration architecture is too simple. In the constructive research on the U-QASAR data integration adapters it was found that the implementation of an adapter for the current version of the platform does not take long for a software developer, if the metrics are known. However, this sets a restriction on developing the adapters: they cannot be universal and target system specific, but rather require special knowledge on the metrics that are to be implemented. Furthermore, they may need to include project-specific calculations if the formula editor of the U-QASAR platform does not provide the calculation methods that the quality monitoring program needs. This sets a need to further develop the data integration protocols.

7.4 Validity and reliability

This section discusses the validity and reliability of this study and provides viewpoints to the threats that should be taken into account when reading this work. The selected schema for the validity discussion is adopted from the work of Runeson and Höst (2009). They propose that in flexible design studies, similar topics for categorization of the discussion perspectives should be used as in controlled experiments. These perspectives are *construct validity*, *internal validity*, *external validity* and *reliability*. In this work the internal validity is not applicable, as this work is mainly explorative and descriptive and does not present causal relationships.

The validity perspectives are discussed as follows: first, Section 7.4.1 describes and discusses the construct validity and then Section 7.4.2 moves on to the external validity. Finally, Section 7.4.3 describes and discusses the reliability of the study. Moreover, threats to the validity relevant in this work are discussed for each of the perspectives. Brink (1993) presents that the main categories for threats to validity and reliability in qualitative research are the researcher, the subjects participating in the project, the situation or social context and the methods of data collection and analysis.

7.4.1 Construct validity

The construct validity refers to the construction of the study; how the results and conclusions reflect the studied objective (Runeson and Höst, 2009). In other words, it discusses how the selection, implementation and use of the research methods and instruments contribute to the attempts of answering the research questions.

To increase the construct validity, appropriate research methods and instruments were selected based on literature. Two research methods were used: the multiple case study method and the constructive research approach. As described in Section 4, both of these are applicable for the research in this thesis, which includes both an explorative and a creative part. Furthermore, suitable research instruments were selected for these methods by the recommendations in literature; for a multiple case study, observations, interviews and questionnaires were used, as suggested by Runeson and Höst (2009) and Maimbo and Pervan (2005). In constructive research, observations and interviews were used as proposed by Lukka (n.d.).

A challenge in this study was that the individual research instruments varied from case to case: many different questionnaires and interview question sets were used, which made it rather difficult to compare the results directly with each other. Furthermore, the first QOS workshop was organized before this study was started and the researcher could not contribute to what data was collected then.

To decrease this instrumental threat to the construct validity, triangulation of the instruments was used in the case study and thematic analysis was used to analyze the qualitative material collected in the case study. Moreover, two peer researchers who had strong backgrounds in qualitative research on software development conducted the data collection in the first workshop. Finally, in the quantitative part of the study, all the scaling questions had a constant scale from 1 to 7 in the questionnaires.

As the use of the QOS methods was studied by two workshops, it was seen that the results from them would be more valid if researchers facilitated the workshops. This ensured a rather similar structure of the event and interpretation of the U-QASAR methodology. The facilitation of the second workshop was enhanced by the feedback on the first workshop, which enabled observing how the enhancements affected the outcome and feedback of the workshop.

Social threats to the construct validity are present in this study. Wohlin et al. (2000) present three forms of social threats for construct validity: hypothesis guessing, evaluation apprehension and experimenter expectancies. All of these may apply to this study. Hypothesis guessing means trying to guess the "correct answer" to a question in a questionnaire or interview and evaluation apprehension means that the experimenters try to look better when they are evaluated (Wohlin et al., 2000). These are difficult to find out about and it is possible that they happened in this study. Experimenters may also have had different expectations of the research, as the members in projects A, B, D and E included members of the U-QASAR project. Furthermore, they might have been biased of their own work and evaluated the methodology and platform to be better than they actually were.

7.4.2 External validity

External validity means the generalizability of the results of the study (Wohlin et al., 2000). In other words, it discusses how the results of the study can be applied in other environments. Several findings in this study would apply to other contexts similar to the five case projects explored in this study, i.e. software development projects having a couple of experienced European software developers with different backgrounds in software quality assurance. Furthermore, the constructive research done on the adapter implementation for the U-QASAR platform can be generalized to apply on other adapter implementations for it. This is further confirmed by that the results were similar to what the existing implementations and the interviews with the other adapter developers revealed.

Wohlin et al. (2000) divides the threats to external validity to three categories: the interaction of selection and treatment meaning the representativeness of the selected cases for the population that the study is wanted to be generalized to, the interaction of

setting and treatment meaning the representativeness of the used practice for the commonly used practices and the interaction of history and treatment meaning the appropriateness of the timing for the research to achieve correct results.

The U-QASAR methodology and platform are based on practices and tools found in the literature, which decreases the threat of setting and treatment. The history angle was not studied in this research, and thus it cannot be discussed further. However, it can be noted that as many of the members of the case projects were also members in the U-QASAR projects and participated in both the first and the second workshop, it may have given them a different view to software quality than a regular software developer.

The case studies represented well industrial software development projects and teams, which decreases the threat of interaction of selection and treatment. However, only five cases were included in the study and the observation time was quite short, which sets certain limits to the findings. The outcome may have a different content in different environments but also similarities should appear. For example, the benefits and challenges of quality monitoring are expected to have similar characteristics. Furthermore, the challenges found in this study should be taken into account in further research.

7.4.3 Reliability

Evaluating the reliability of the study means the discussion about the dependability of the research outcome on the researcher (Runeson and Höst, 2009), i.e. how a specific researcher has affected the outcome of the research by drawing conclusions from the collected data. The reliability can also be seen as how well the same outcome could be achieved by reproducing the study (Runeson and Höst, 2009).

This study was conducted by one researcher, which means that the conclusions are only based on her thinking. Despite that the researcher in this study made an extensive literature review on the topic of software quality and software measurement, she was not an expert in software quality assurance and the learning on the topic happened through the research project. She had not majored in computer science, which would have given a more solid background on software development and issues related to software quality. However, she had an academic background by majoring in automation and systems technology and her studies also included programming and software engineering.

To increase the reliability of this study, the repeatability and reproducibility were promoted by documenting the used instruments and analysis methods as accurately as possible. However, it has to be kept in mind that as the results of interviews and questionnaires depend on the respondents' character and state of mind at the moment of responding and the observations heavily depend on the researcher's character and experience, similar results as in this study may not be achieved in a reproduced research.

8 Conclusions

This section provides an overview of the findings in this thesis. First, Section 8.1 presents the summary of this study and then in Section 8.2 suggestions for the future research are provided.

8.1 Summary

This study explored how a quality monitoring methodology and platform could be utilized in industrial software development. The study included a case study and constructive research. The case study was used to understand the study objective from the practitioner's point of view and the constructive research helped to understand the architecture of the data integration in the current version of the U-QASAR platform.

Software quality is a complex concept and there have been various studies to explore it. The literature is wide but fragmented, and a common terminology has not been established. This thesis contributed in the literature by providing a review on the concepts of quality models and software measurement. Moreover, it provided new research material by conducting a multiple case study on designing a quality monitoring program and deploying it in industrial software development projects.

Monitoring quality in software projects helps to guide the project and to make management decisions. Quality should be defined with project-specific quality models, which provide a base for a quality monitoring program. This thesis strengthened the existing theory with many findings, such as: 1) quality can be defined by the GQM paradigm, 2) connecting quality metrics to quality objectives is difficult and 3) interpretation challenges are possible when monitoring quality. Furthermore, the results include challenges that should be taken into account while deploying quality monitoring.

The explored U-QASAR platform provides quality monitoring by software measurement data integration and this study found it to be a cost-effective practice. However, it was also found that the current version of the platform does not support the data integration in an effective way, which could cause challenges for keeping it in use.

8.2 Future work

Five matters can be seen to create the base on the further research around the topic and the content of this thesis. These are 1) the extension of the case study research, 2) the enhancement of the U-QASAR methodology, 3) the enhancement of the U-QASAR platform, 4) research on all the existing quality monitoring tools instead of only those presented in literature and 5) creating brand awareness around the U-QASAR concept.

The multiple case study method in this thesis was rather limited and the duration of the observation phase was relatively short. Thus, to provide more reliable results, the study should be extended by the amount of case projects, the diversity of different sizes of case projects, the diversity of stakeholders and the duration and intensity of observation.

The architecture of the U-QASAR platform, especially concerning the data integration, should be revised to take it to a more generalized direction. As the current situation of the integration approach is a data mapping which might have to be implemented differently for different projects because of varying configurations, the possibility to reuse of the adapters is not maximized. This should be studied first by reviewing the current proposals for data integration in literature, and then by developing a more general solution. A possible starting point for the research could lay in semantic data integration proposed by the DataBearings project in Teknologian tutkimuskeskus VTT Oy ("DataBearings," n.d.).

The other branch for the further development of the U-QASAR platform is the enhancement and addition of the features that it provides. Currently, the possibility for statistical analysis in the platform is limited. It was proposed by Project E that the statistical analysis could be added to the platform by integrating a toolset such as R or Matlab. As was revealed by the literature review in Section 2, this has already been attempted to be implemented in another quality monitoring tool. Thus, it seems as a recommendable addition to the U-QASAR platform as well. However, if different stakeholders use the U-QASAR platform, also the option for more simple analysis should be available in the platform.

Another feature suggestion for the platform is a decision support system to enhance the decision making process based on the provided data, as also suggested by Project E. Enhanced mobile services and gamification proposed by Project A could increase the interest towards the platform. Gamification could further enable the measurement incentives mentioned in the success factors for deploying a quality monitoring program.

Another feature proposal was identified by observation in internal discussions in the project. Currently, the U-QASAR platform provides information about the changes in the produced data. However, if the cause of a certain change was identified but not fixed immediately, the data changes by this cause should be able to be hidden or ignored automatically as the problem identification always consumes resources.

As discussed in Section 7.1, the U-QASAR methodology should be further developed to provide more practical guidelines for the whole process of quality enhancement from the definition phase to the process improvement. These guidelines should also be validated by empirical research. Based on the long history of the research of software quality improvement, this part of the future work seems to be the most difficult to implement successfully.

Finally, as mentioned in Section 7.2, the permanence of software quality monitoring tools in the literature seems not to be very good. Furthermore, the visibility and reachability of the in-house tools is limited. It would be relevant to do a market review of the available commercial tools to support or revoke the idea in this thesis that no overriding quality monitoring tool has established its status in the quality assurance community. Effort should also be put into keeping the U-QASAR platform in the offering of quality monitoring tools. This would probably be easiest done by open source licensing of the tool, which could lead to more extensive further development and deployment of it in the open source community.

Bibliography

Alandes, M., Kenny, E.M., Meneses, D., Pucciani, G., 2012. Experiences with Software Quality Metrics in the EMI middleware. J. Phys. Conf. Ser. 396, 052003. doi:10.1088/1742-6596/396/5/052003

Al-Kilidar, H., Cox, K., Kitchenham, B., 2005. The use and usefulness of the ISO/IEC 9126 quality standard, in: Empirical Software Engineering, 2005. 2005 International Symposium on. IEEE, p. 7–pp.

Asato, R., De Mesquita Spinola, M., Costa, I., De Farias Silva, W.H., 2009. Alignment between the business strategy and the software processes improvement: A roadmap for the implementation, in: PICMET: Portland International Center for Management of Engineering and Technology, Proceedings. pp. 1066–1071. doi:10.1109/PICMET.2009.5262035

Basili, V., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., Seaman, C., Trendowicz, A., 2007. Bridging the gap between business strategy and software development. ICIS 2007 Proc. 25.

Basili, V., R., Weiss, D., M., 1984. A Methodology for Collecting Valid Software Engineering Data 10.

Berry, M., Jeffery, R., 2000. An instrument for assessing software measurement programs. Empir. Softw. Eng. 5, 183–200.

Berry, M., Vandenbroek, M.F., 2001. A Targeted Assessment of the Software Measurement Process. IEEE Computer Society.

Bhatti, A.M., Abdullah, H.M., Gencel, C., 2009. A model for selecting an optimum set of measures in software organizations, in: Software Process Improvement. Springer, pp. 44–56.

Boehm, B.W. (Ed.), 1978. Characteristics of software quality, TRW series of software technology. Amsterdam ; New York.

Börjesson, A., 2006. Simple Indicators for Tracking Software Process Improvement Progress, in: Richardson, I., Runeson, P., Messnarz, R. (Eds.), Software Process Improvement, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 74–87.

Brandtner, M., Giger, E., Gall, H., 2014. Supporting continuous integration by mashing-up software quality information, in: Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on. IEEE, pp. 184–193.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3, 77–101. doi:10.1191/1478088706qp063oa

Briand, L.C., Differding, C.M., Rombach, H.D., 1996. Practical guidelines for measurement-based process improvement. Softw. Process Improv. Pract. 2, 253–280.

Brink, H.I.L., 1993. Validity and reliability in qualitative research. Curationis 16, 35–38.

CISQ, 2012. CISQ Specifications for Automated Quality Characteristic Measures.

Coman, I.D., Sillitti, A., Succi, G., 2009. A case-study on using an Automated In-process Software Engineering Measurement and Analysis system in an industrial environment, in: IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009. Presented at the IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009, pp. 89–99. doi:10.1109/ICSE.2009.5070511

DataBearings [WWW Document], n.d. URL https://sites.google.com/site/databearings/home (accessed 12.13.15).
Deissenboeck, F., Juergens, E., Hummel, B., Wagner, S., Pizka, M., others, 2008. Tool support for continuous quality control. Softw. IEEE 25, 60–67.
Deissenboeck, F., Juergens, E., Lochmann, K., Wagner, S., 2009. Software quality models: Purposes, usage scenarios and requirements, in: Software Quality, 2009. WOSQ'09. ICSE Workshop on. IEEE, pp. 9–14.
Dekkers, C.A., McQuaid, P.A., 2002. The dangers of using software metrics to (mis) manage. IT Prof. 24–30.
Dowson, M., 1993. Software process themes and issues, in: Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the. IEEE, pp. 54–62.
Dromey, G.R., 1995. A model for software product quality. Softw. Eng. IEEE Trans. On 21, 146–162.
Dromey, R.G., 1998. Software product quality: Theory, model and practice. Softw. Qual. Inst. Brisb. Aust.
Emam, K.E., 2005. The ROI from Software Quality: An Executive Briefing. Auerbach Publications.
Fenton, N., 1994. Software measurement: A necessary scientific basis. Softw. Eng. IEEE Trans. On 20, 199–206.
Fonseca, V.S., Barcellos, M.P., de Almeida Falbo, R., 2015. Integration of Software Measurement Supporting Tools: A Mapping Study.
Frederiksen, H.D., Iversen, J., 2003. Implementing Software Metrics Programs: A Survey of Lessons and Approaches.
Fuggetta, A., Lavazza, L., Morasca, S., Cinti, S., Oldano, G., Orazi, E., 1998. Applying GQM in an industrial software factory. ACM Trans. Softw. Eng. Methodol. TOSEM 7, 411–448.
Garvin, D., A., 1984. What Does "Product Quality" Really Mean? MIT Sloan Manag. Rev. 26, 25–43.
Gopal, A., Krishnan, M.S., Mukhopadhyay, T., Goldenson, D.R., 2002. Measurement programs in software development: determinants of success. Softw. Eng. IEEE Trans. On 28, 863–875.
Grady, R.B., 1992. Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
Hofman, R., 2011. Behavioral economics in software quality engineering. Empir. Softw. Eng. 16, 278–293. doi:10.1007/s10664-010-9140-x
IEEE, 1998. 1061-1998 - IEEE Standard for a Software Quality Metrics Methodology.
ISO/IEC, 2010. ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models.
ISO/IEC, 2001. ISO/IEC 9126 Software engineering — Product quality.
Jørgensen, M., 1999. Software quality measurement. Adv. Eng. Softw. 30, 907–912.
Kaner, C., Bond, W.P., 2004. Software Engineering Metrics: What Do THey Measure and How Do We Know?
Kaplan, R., Norton, D., 1992. The balanced scorecard--measures that drive performance. Harv. Bus. Rev. 70, 71 –79.

Karch, E., 2011. The Software Crisis: A Brief Look at How Rework Shaped the Evolution of Software Methodolgies - Karchworld Identity - Site Home - MSDN Blogs [WWW Document]. URL http://blogs.msdn.com/b/karchworld_identity/archive/2011/04/04/the-software-crisis-how-methodologies-evolved-from-the-influence-of-rework.aspx (accessed 12.21.15).

Kasanen, E., Lukka, K., Siitonen, A., 1993. The Constructive Approach in Management Accounting Research. J. Manag. Account. Res. 5, 243–264.

Kilpi, T., 2001. Implementing a software metrics program at Nokia. Softw. IEEE 18, 72–77.

Kitchenham, B., Pfleeger, S.L., 1996. Software quality: the elusive target [special issues section]. IEEE Softw. 13, 12–21. doi:10.1109/52.476281

Lukka, K., n.d. The Constructive Research Approach.

Madeyski, L., Majchrzak, M., 2014. Software Measurement and Defect Prediction with Depress Extensible Framework. Found. Comput. Decis. Sci. 39. doi:10.2478/fcds-2014-0014

Maimbo, H., Pervan, G., 2005. Designing a case study protocol for application in IS research. PACIS 2005 Proc. 106.

McCall, J.A., Richards, P.K., Walters, G.F., 1977. Factors in software quality. DTIC Document.

Münch, J., Abrahamsson, P. (Eds.), 2007. Product-focused software process improvement: 8th International conference, PROFES 2007, Riga, Latvia, July 2-4, 2007: proceedings, Lecture notes in computer science. Springer, Berlin ; New York.

Niessink, F., Van Vliet, H., 2001. Measurement program success factors revisited. Inf. Softw. Technol. 43, 617–628.

Offen, R.J., Jeffery, R., 1997. Establishing software measurement programs. IEEE Softw. 14, 45–53. doi:10.1109/52.582974

O'Regan, G., 2014. Introduction to Software Quality, Undergraduate Topics in Computer Science. Springer International Publishing, Cham.

Parkinson, S.T., Counsell, S., Norman, M., Hierons, R.M., Lycett, M., 2008. The precursor to an industrial software metrics program, in: Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on. IEEE, pp. 221–226.

Parkinson, S.T., Hierons, R.M., Lycett, M., Norman, M., 2010. Practitioner-based measurement: a collaborative approach. Commun. ACM 53, 142. doi:10.1145/1666420.1666456

Park, R.E., Goethert, W.B., Florac, W.A., 1996. Goal-driven software measurement - A guidebook.

Pressman, R.S., 2010. Software engineering: a practitioner's approach. McGraw-Hill Higher Education, New York.

Rombach, H.D., Ulery, B.T., 1989. Improving software maintenance through measurement. Proc. IEEE 77, 581–595.

Rowe, A., Whitty, R., 1993. Ami: promoting a quantitative approach to software management. Softw. Qual. J. 2.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164. doi:10.1007/s10664-008-9102-8

Smart, J., 2011. Jenkins - The definitive guide. O'Reilly Media, Inc.

Sommerville, I., 2011. Software engineering. Pearson, Boston.

Southekal, P.H., 2014. Implementing the Stakeholder based Goal-Question-Metric (GQM) Measurement Model for Software Projects. Trafford Publishing.
Tahir, T., Gencel, C., 2010. A structured goal based measurement framework enabling traceability and prioritization, in: Emerging Technologies (ICET), 2010 6th International Conference on. IEEE, pp. 282–286.
Tahir, T., Jafar, A., 2011. A Systematic Review on Software Measurement Programs. IEEE, pp. 39–44. doi:10.1109/FIT.2011.15
US Dept. of Defense and US Army, 2006. Practical Software and Systems Measurement (PSM) - Methods of Operation.
Van Latum, F., Van Solingen, R., Oivo, M., Hoisl, B., Rombach, D., Ruhe, G., 1998. Adopting GQM-based measurement in an industrial environment. IEEE Softw. 78–86.
Van Solingen, R., Berghout, E., 2001. Integrating goal-oriented measurement in industrial software engineering: industrial experiences with and additions to the Goal/Question/Metric method (GQM), in: Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International. IEEE, pp. 246–258.
Wagner, S., 2013. Software Product Quality Control. Springer Berlin Heidelberg, Berlin, Heidelberg.
Wagner, S., 2007. Using economics as basis for modelling and evaluating software quality, in: Proceedings of the First International Workshop on The Economics of Software and Computation. IEEE Computer Society, p. 2.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell, MA, USA.
Yin, R.K., 2014. Case study research : design and methods. SAGE, Los Angeles :

Appendices

Appendix A Data integration tools

Table 22. Software measurement tools in literature. An empty field means that the information was not available

Name		SQuAVisiT	ConQAT	Alitheia Core	SPDW+	SOFAS	3C	QualitySpy
Year		2007	2008	2009	2010	2011	2012	2012
Type		Toolset	Platform	Platform	Framework	Platform	Approach	Tool
Target users		Developers	All stakeholders			All stakeholders	Developers	
Availability			Download	Download		Contact information		Download
Data	Integration	Yes		Yes	Yes	Yes	Yes	Yes
	Visualization	Yes					Yes	
	Reporting						Yes	Yes
Focus		Quality assessment		Quality	Quality	Analysis result integration	Quality	Uniform data collection
Client				Browser/ Standalone	Browser	Browser	Browser	Browser
Data source integration		Yes		Yes	Yes	Yes	Yes	Yes
Statistical support								
DSS support								
GQM-based				Yes	Yes			Yes
Quality modeling				Yes				
Project support				Yes	Yes	Yes	Yes	Yes
Platform customization				Yes		Yes		

Name		Dione	UQM	ASSIST	DePress	SQA-Mashup	U-QASAR
Year		2012	2012	2013	2014	2015	2015
Type		Tool	Application	Tool	Platform	Platform	Platform
Target users		Developers, management		Managers	QA, SPI,Management	All stakeholders	All stakeholders
Availability				In-house	Download	Download	Download
Data	Integration	Yes	Yes	Yes	Yes	Yes	Yes
	Visualization	Yes		Yes	Yes	Yes	Yes
	Reporting	Yes		Yes	Yes		Yes
Focus		Defect prediction Quality control	Quality monitoring	Measurement	Defect prediction	Data integration	Quality
Client		Browser		Standalone	Standalone	Browser	Browser
Data source integration		Yes	Yes	Yes	Yes	Yes	Yes
Statistical support					Yes		
DSS support							
GQM-based			Yes	Yes	-		Yes
Quality modeling			Yes	Yes			Yes
Project support		Yes		Yes	Yes	Yes	Yes
Platform customization					-	Yes	Yes

Appendix B Quality objective elaboration template

Figure 17. The quality objective template that was used for elaborating quality objectives in the workshop.

INDICATOR 1

Name: low response times		
Target levels	Lower breakpoint:	
	Higher breakpoint:	
	Target level:	
Frequency	Measuring interval:	
Data	Easiness of collecting:	
Metrics	Metric 1	
	Metric 2	
	Metric 3	
	Metric 4	
	Metric 5	
	Metric 6	

VOTING: How descriptive is this indicator for the project?

Name	**Vote**				
	1	**2**	**3**	**4**	**5**

Scale:
1 - Not descriptive at all
2
3
4
5 - Very descriptive

VOTING: How easy is this indicator to implement?

Name	**Vote**				
	1	**2**	**3**	**4**	**5**

Scale:
1 - Very difficult
2
3
4
5 - Very easy

NOTES

Appendix C Questionnaires

Pre-questionnaire
Projects A, B and C

1. What is your name?
2. [Only A] What is your workshop number?
   - ○ Group 1
   - ○ Group 2
   - ○ Group 3
3. Express the roles that describe your experience the best
   - ☐ Customer
   - ☐ Sales representative
   - ☐ Requirements engineer
   - ☐ Project manager
   - ☐ Quality assurance
   - ☐ Product manager
   - ☐ Software architect
   - ☐ Programmer
   - ☐ Tester
   - ☐ Other:
4. Your industrial software development experience in years is...
   - ○ 0-1 years
   - ○ 1-3 years
   - ○ 3-5 years
   - ○ 5-10 years
   - ○ 10-20 years
   - ○ Over 20 years
5. Have you done quality assurance previously?
   - ▪ Never before 1…7 Very often
6. Have you defined quality objectives previously?
   - ▪ Never before 1…7 Very often
7. How have you defined quality objectives previously?
   - ☐ Individually (I have done these kind of tasks alone)
   - ☐ Collaboratively (I have done these kind of tasks as a group work)
8. Have you defined quality indicators or metrics previously?
   - ▪ Never before 1…7 Very often
9. How have you defined quality indicators and metrics previously?
   - ☐ Individually (I have done these kind of tasks alone)
   - ☐ Collaboratively (I have done these kind of tasks as a group work)
10. Have you used quality objectives, indicators, or metrics previously?
    - ▪ Never before 1…7 Very often
11. Please, explain how you have used quality objectives, indicators, or metrics.
12. What quality objectives are relevant for U-QASAR software tool? Please, name and shortly describe FIVE quality objectives at most.
13. [Only A] How well do you know the U-QASAR software tool?
    - ▪ Very well 1…5 Never even tried

Figure 18. Pre-questionnaire for projects A, B and C.

Post-questionnaire
Project A

Your background information

1. What is your name?

Workshop outcome

2. The usefulness of the outcome of today's workshop for U-QASAR project is...
   - Very low 1…7 Very high
3. The coverage of the outcome of today's workshop for U-QASAR project is...
   - Very low coverage 1…7 Very high coverage

Workshop participants

4. Who should participate to the definition of quality objectives, indicators, and metrics?
   - ☐ Customer
   - ☐ Sales representative
   - ☐ Requirements engineer
   - ☐ Project manager
   - ☐ Quality assurance
   - ☐ Product manager
   - ☐ Software architect
   - ☐ Programmer
   - ☐ Tester
   - ☐ Other:

Workshop method

5. Did you find the workshop method effective in making you to contribute?
   - Not at all 1…7 Very much
6. How easy it was to share important knowledge over the workshop participants?
   - Not easy at all 1…7 Very easy indeed
7. In comparison to the practices you have used earlier, the cost-efficiency of today's workshop for defining quality objectives is...
   - Very low 1…7 Very high
8. In comparison to the practices you have used earlier, the ease-of-use of today's workshop for defining quality objectives is...
   - Very low 1…7 Very high
9. The usefulness of today's workshop method for software projects in general is...
   - Very low 1…7 Very high
10. How much you would like to use this kind of a workshop in your other software projects...
    - Not at all 1…7 Very much
11. What could be improved in today's workshop method?
12. What worked well in today's workshop method?

Software tool

13. Evaluate the general ease-of-use of the U-QASAR software tool
    - Very low 1…7 Very high
14. Evaluate the learnability of the U-QASAR software tool
    - Very low 1…7 Very high
15. How much did the workshop method and U-QASAR software tool supported one another?
    - Not at all 1…7 Very much
16. Did you recognize any important quality knowledge that could not be entered to the tool?
    - Everything fits in 1…7 Nothing fits in
17. Evaluate the quality of the implemented functionality in the U-QASAR software tool
    - Very low 1…7 Very high
18. What could be improved in the current version of the U-QASAR software tool?
19. What worked well in the current version of the U-QASAR software tool?

Figure 19. Post-questionnaire for Project A.

Workshop post-questionnaire
Project B and Project C

Your background information

1. What is your name?

Workshop outcome

2. How useful was today's workshop for your project?
   - Not useful at all 1…7 Very useful
3. How covering was the outcome of the workshop regarding QUALITY OBJECTIVES?
   - Not covering at all 1…7 Very well covering
4. How covering was the outcome of the workshop regarding QUALITY INDICATORS?
   - Not covering at all 1…7 Very well covering
5. How covering was the outcome of the workshop regarding QUALITY METRICS?
   - Not covering at all 1…7 Very well covering
6. How covering was the outcome of the workshop AS A WHOLE?
   - Not covering at all 1…7 Very well covering
7. Do you find that something is missing from the outcome? Please justify your answer.
8. Do you find the workshop outcome ready-to-use?
   - Nothing is ready-to-use 1…7 Everything is ready-to-use
9. How many of the results require a multiple-source data integration tool for monitoring them?
   - None 1…7 All

Workshop participants

10. Who should participate to the definition of quality objectives, indicators and metrics?
    - ☐ Customer
    - ☐ Sales representative
    - ☐ Requirements engineer
    - ☐ Project manager
    - ☐ Quality assurance
    - ☐ Product manager
    - ☐ Software architect
    - ☐ Programmer
    - ☐ Tester
11. Other:
12. Do you find that the number of the participants was enough for the workshop?
    - ○ Not enough at all
    - ○ Not enough
    - ○ Just the right amount
    - ○ Slightly too many
    - ○ All too many

Workshop method

13. Did you find the workshop method effective in making you to contribute?
    - Not effective at all 1…7 Very effective
14. How easy was it to share important knowledge among the workshop participants?
    - Not easy at all 1…7 Very easy
15. Please evaluate the COST-EFFICIENCY of Friday's workshop for defining a quality model.
    - Very low 1…7 Very high
16. Please evaluate the EASE-OF-USE of Friday's workshop for defining a quality model.
    - Very low 1…7 Very high
17. In general, how useful is today's workshop method for other software projects?
    - Not useful at all 1…7 Very useful
18. What could be improved in today's workshop method?
19. What worked well in today's workshop method?
20. Would you like to use this kind of a workshop in your other software projects?

Figure 20. Post-questionnaire for Project B and Project C.

Quality model definition post-questionnaire
Projects D and E

Background information

1. What is your name?
2. Company name:
3. Express the roles that best describe your experience
   - ☐ Customer
   - ☐ Sales representative
   - ☐ Requirements engineer
   - ☐ Project manager
   - ☐ Quality assurance
   - ☐ Product manager
   - ☐ Software architect
   - ☐ Programmer
   - ☐ Tester
   - ☐ Other:

Methods

4. How was the quality model constructed?
5. Did you use the provided instructions for creating the quality model?
   - Scale: No instructions were followed 1…7 All the instructions were followed
6. Were the provided instructions useful?
   - Not useful at all 1…7 Very useful
7. Did you use the provided handouts as a support for creating the quality model?
   - Nothing was used 1…7 All the materials were used
8. Were the provided handouts useful?
   - Not useful at all 1…7 Very useful

Quality model

9. How covering are the created quality objectives?
   - Not covering at all 1…7 Very well covering
10. How covering are the created quality indicators?
    - Not covering at all 1…7 Very well covering
11. How covering are the created quality metrics?
    - Not covering at all 1…7 Very well covering
12. How covering is the quality model as a whole?
    - Not covering at all 1…7 Very well covering

Use of the quality model

13. How useful is the created quality model for your project?
    - Not useful at all 1…7 Very useful
14. Do you find the defined quality model ready-to-use?
    - Nothing is ready-to-use 1…7 Everything is ready-to-use
15. How many parts of the created quality model require a multiple-source data integration tool for monitoring them?
    - None 1…7 All
16. Do you see it possible to construct a quality model in the beginning of every project?
    - Not for any project 1…7 For all projects
17. Why?
18. Do you find that something is missing from the created quality model? Please justify your answer.

Quality objective setting method

19. Is it useful to describe the quality of a development process by quality models?
    - Not useful at all 1…7 Very useful
20. Is it useful to describe the quality of a development project outcome by quality models?
    - Not useful at all 1…7 Very useful
21. Do you know some other method that is more effective in defining quality models? Please describe below.

Figure 21. Quality model definition post-questionnaire for projects D and E

Project background information form
Project B and Project C

1. Your name
2. Company name
3. Project name
4. What is the start date for the project?
5. What is the scheduled end date of the project?
6. Project description
7. What is the process type of the project?
   - ○ Waterfall
   - ○ Agile / Lean
   - ○ Prototyping
   - ○ Rapid development
   - ○ Other:
8. What is the financial size of the project
   - ▪ Very small 1…7 Very big
9. How is the end product of the project used?
   - ○ Internally
   - ○ Sold to and external company
   - ○ Free of use
10. Does the project have a previously created quality mode
    - ○ Yes
    - ○ No

Participants of the project

11. How many people are working in the project?
12. Project stakeholders
13. How experienced are the participants of the project?
    - ▪ Not experienced at all 1…7 Very experienced
14. U-QASAR users

Using U-QASAR

15. What is intended to be achieved by using U-QASAR?
16. What is intended to be monitored by using U-QASAR?
17. How frequently is it intended to use U-QASAR?

Testing organization

18. How large is the testing organization behind the project?
    - ▪ Very small (<2 people) 1…7 Very large (>20 people)
19. How is the testing done for the project?
    - ○ After each iteration (iteration < 1 month)
    - ○ After each iteration (iteration > 1 month)
    - ○ In the end of the project
20. Anything else?

Figure 22. Project background information form for Project B and Project C.

Project background information form
Projects D and E

1. Your name
2. Company name
3. Project name
4. What is the start date for the project?
5. What is the scheduled end date of the project?
6. Project description
7. What is the process type of the project?
   - ☐ Waterfall
   - ☐ Agile/Lean
   - ☐ Prototyping
   - ☐ Rapid development
   - ☐ Other:
8. Who is the client of the project?
   - ○ Internal client
   - ○ External client
   - ○ No client, free of charge
9. What kind of existing quality documentation is there for the project, if any?

Participants of the project

10. How many people are working on the *project?
11. Project stakeholders
12. How experienced are the participants of the project?
    - Not experienced at all 1…5 Very experienced
13. U-QASAR users

Using U-QASAR

14. What is intended to be achieved by using U-QASAR?
15. What is intended to be monitored by using U-QASAR?
16. How frequent is the intended use of U-QASAR?

Figure 23. Project background information for projects D and E.

Follow-up questionnaire
Projects D and E

Background

1. What is your name

Using U-QASAR

2. Have you used the U-QASAR platform?
    - No, I haven't used it 1…7 Yes, very often
3. How many times have you used the U-QASAR platform?
4. Why did you use the U-QASAR platform?
5. How did you use the U-QASAR platform?
6. How do you see the usability of the U-QASAR platform?
    - Not good at all 1…7 Very good
7. Comments on previous answer:
8. In what kinds of situations do you think the U-QASAR platform could be used?
9. How much time would you be able to use for updating manual metrics on the platform?

The information provided by the platform

10. Did the U-QASAR platform provide tou any useful information?
    - Not useful information at all 1…7 A lot of useful information
11. Comments on previous answer:
12. Do you think you would be able to make decisions based on the information provided by the U-QASAR platform?
13. What is useful quality information from your point of view?

Timing

14. How much time did it take for you to understand how the U-QASAR platform works?
    - A lot of time 1…7 Very small amount of time
15. Comments on previous answer:
16. How do you see the time that is needed for using the U-QASAR platform?

Motivation

17. What things did or would motivate you to use the U-QASAR platform?
18. What thing did or would discourage you to use the U-QASAR platform?
19. What do you think is the most useful characteristic(s) of the U-QASAR platform?
20. What kind of benefits do you see in using the U-QASAR platform?
21. Would you like to take the U-QASAR platform into use in your other projetcs?
    - ○ Yes
    - ○ Yes, if some things were changed
    - ○ Maybe
    - ○ No
    - ○ I don't know
    - ○ Other:
22. Comments on previous answer:

Future

23. Is there something that you would change in the U-QASAR platform to make it more satisfying?

Additional comments

24. Free comments

Figure 24. The follow-up questionnaire for projects D and E.

Final questionnaire
Projects B, C, D and E

1. Project description

2. Results
In this section, the results and experiences gained in the U-QASAR assessment project are described. This section is divided into three parts. First, the execution of the project and U-QASAR piloting activities are described in section 2.1. Second, the experienced benefits and improvements of the use of U-QASAR in the pilot project are presented in section 2.2, followed by the analysis of the experienced challenges and developed solutions in section 2.3. Finally, the recommendations for improvements and further development of U-QASAR are outlined in section 2.4.

2.1 Execution of the project
2.1.1 Quality models

2.2 Experienced benefits

2.3 Challenges and solutions

2.4 Recommendations for further development

Figure 25. The final questionnaire for projects B, C, D and E. In this questionnaire, the practitioners could write their experiences themselves. This approach was used to confirm the researchers' conclusions from the other questionnaires and the interviews.

Appendix D Follow-up interview questions

Follow-up interview 1
Project B and Project C

Usage

1. Have you used U-QASAR?
    a. How many times have you used it?
    b. Why have you used it?
    c. How have you used it?
        i. What different things did you do with it?
        ii. What features did you use?
    d. (Why have you not used it?)
2. Do you think it was easy to use U-QASAR?
3. What do you think about the manual versus automated metrics?
    a. How much time would you give for updating a quality model?

Information

4. Did U-QASAR provide you some useful information?
    a. What information?
    b. Why not?
5. Do you think you would be able to make decisions based on the information provided by U-QASAR?
6. What would be useful quality information from your viewpoint?
    a. In what form?
    b. What kinds of metrics?
    c. What kind of visualization?
    d. With what frequency?

Timing

7. How long did it take to understand
    a. how the models work?
    b. how the model data can be used?
8. How do you see the amount of time that has to be used for using U-QASAR?

Motivation

9. What did (would) motivate you to use U-QASAR?
    a. What things could you name that would encourage you to use it?
    b. What did (would) discourage you to use U-QASAR?
10. What do you think is the most useful characteristic of U-QASAR?

Future

11. What would you change in U-QASAR to make it more satisfying?

Figure 26. The interview questions in the first follow-up interviews with Project B and Project C. The questions in the second interview were only slightly different, having some modification to the words used as it was not the first meeting.

Follow-up interview 3
Project B

1. What kind of benefits do you see in using the U-QASAR?
    a. for you?
    b. for the project?
    c. for the team?
    d. for the company?

2. What kind of problems or challenges do you see in using the U-QASAR?
    a. for you?
    b. for the project?
    c. for the team?
    d. for the company?

3. If you look at the model in the platform now, how well do you think it represents your part of the project?
    a. Does it include relevant things?
    b. Is something missing? What?

4. How is quality present in the current situation?
    a. Is there something could be added to the platform so it would be useful at the moment when there is no development?

5. For what purposes would you use the current platform and the information it provides?

6. What kind of projects should use U-QASAR?
    a. Do you think projects of all sizes could use U-QASAR?

Figure 27. Interview questions in the third follow-up interview with Project B. It was not meaningful to continue with the old questions, as the project was not able to use the platform. The interviews were disrupted after this.

Appendix E Final interview questions

Final interview
Projects D and E

1. So you have described that you created the quality model on the basis of the existing U-QASAR model?
    a. What equipment did you use to create the quality model? Did you do it in the U-QASAR platform?
    b. Did you decide the threshold limits at the same time?
    c. Who created the model?
    d. What challenges/risks do you see in the process of creating the model?
2. How do you see the process of deploying a quality measurement program when you have already defined what quality means for your project?
    a. What was done?
    b. Was there training for the users?
    c. Was it smooth?
    d. How much time did it take?
3. Do you see that it was cost-effective to deploy a quality measurement program in a project with the U-QASAR tool?
    a. Do you think that the benefits exceed the costs?
    b. Would you recommend U-QASAR for other companies based on the easiness to take it into use?
4. What challenges/risks do you see in the deployment process?
5. How was the used quality measurement program seen?
    a. How did it respond to your needs for project (quality) information?
    b. Do you find the provided information useful for you?
    c. Was there enough information available to base decision making?
6. Did you modify your quality model during the project?
    a. Do you think it should be monitored and modified continuously by some person or the team?
    b. Do you see that as cost-effective?
7. Did the use of the U-QASAR platform make you think differently about any things in the software development project?
8. How did you feel about using the U-QASAR platform for quality monitoring?
9. How do you see the approach to model quality in the U-QASAR platform?
    a. Objective-indicator-metric tree
    b. Customizable widgets
    c. Overview of the project
10. In which phases of a software development project do you think the U-QASAR platform could be used?
    a. Planning
    b. Requirements engineering
    c. Development
    d. Deployment
    e. Maintenance
11. Do you think that the U-QASAR platform should be used by all the stakeholders of a software development project?
    a. Developers
    b. Managers
    c. Customers
    d. Architects
    e. Users
    f. Quality managers
12. Do you think you would be able to manage your project without other monitoring tools than U-QASAR platform?
13. What are your overall feelings about using U-QASAR?

Figure 28. Questions in the final interview.

Appendix F Adapter interview questions

General structure for the adapter interviews

General

1. What system is the adapter intended for?
2. Were you familiar with this system?
3. Is the adapter ready?
4. What kinds of pre-requisites were there for implementing the adapter?

Programmer

5. How experienced are you as a programmer?
6. Which programming language did you use?
7. Were you familiar with this language?
8. Were you familiar with the programming environment?

Construction

9. Does the adapter include computations?
10. Did U-QASAR add any constraints to the use or implementation of the adapter?
11. Are there any constraints with the adapter?
12. Can the constraints be a problem if a company wants to create an own adapter?

Process

13. How long did it take for you to implement the adapter?
14. Can you tell me about the implementation process?
15. What kinds of challenges were there when implementing the adapter?
16. Did you find any useful shortcuts?
17. Do you have suggestions for me or someone else who is going to implement an adapter?

Testing

18. Have you tested the adapter?

The usage

19. Where is the adapter used now?
20. What metrics have been implemented with the adapter?
21. Have there been any problems or improvement suggestions?

Figure 29. Interview questions in the interviews with the adapter developers. Additional questions were asked in case the researcher saw it necessary.

Appendix G Quantitative data on workshop experience

Table 23. Raw data from the post-questionnaire.

		Did you find the workshop method effective in making you to z?	How easy it was to share important knowledge over the workshop participants?	Please evaluate the cost-efficiency of the workshop method for defining a quality model	Please evaluate the ease-of-use of the workshop method for defining a quality model.	In general, how useful is the workshop method for other software projects?	Would you like to use this kind of a workshop in your other software projects?	Do you see it possible to organize a workshop in the beginning of every project?	How useful was today's workshop for your project?
Aggregated	**Scale**	**N = 19**	**N = 19**	**N = 20**	**N = 20**	**N = 19**	**N = 19**	**N = 5**	**N = 5**
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0	1	2	0	0	2	0
	4	2	3	3	4	1	2	1	0
	5	5	3	8	5	5	6	2	2
	6	6	5	7	6	9	6	0	3
	7	6	8	1	3	4	5	0	0
PROJECT A	**Values**	4	4	4	4	4	4	-	-
		5	6	5	5	5	6	-	-
		7	5	4	4	6	6	-	-
		7	5	6	6	7	7	-	-
		6	7	6	6	7	7	-	-
		6	6	7	6	6	6	-	-
		7	7	5	7	7	7	-	-
		7	7	6	5	6	6	-	-
		5	6	4	3	5	5	-	-
		7	5	6	7	7	7	-	-
		5	7	5	5	6	5	-	-
		6	6	5	5	6	5	-	-
		7	7	5	6	5	7	-	-
		6	4	5	3	5	4	-	-
	Scale								
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0	0	2	0	0	0	0
	4	1	2	3	2	1	2	0	0
	5	3	3	6	4	4	3	0	0
	6	4	4	4	4	5	4	0	0
	7	6	5	1	2	4	5	0	0
	Sum of votes	**14**	**14**	**14**	**14**	**14**	**14**	**0**	**0**
PROJECTS B AND C	**Values**	5	4	6	4	6	6	5	6
		6	7	5	5	6	5	4	6
		5	7	5	6	6	5	3	5
		4	6	3	4	5	5	3	5
		6	7	6	7	6	6	5	6
		-	-	6	6	-	-	-	-
	Scale								
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0	1	0	0	0	2	0
	4	1	1	0	2	0	0	1	0
	5	2	0	2	1	1	3	2	2
	6	2	1	3	2	4	2	0	3
	7	0	3	0	1	0	0	0	0
	Sum of votes	**5**	**5**	**6**	**6**	**5**	**5**	**5**	**5**

Appendix H Quotes on the QOS method

Table 24. Quotes behind the QOS method feedback categories. The high-level concepts identified from the raw data are listed in the left column and the right column includes the number of their appearance in projects A, B, and C.

POSITIVE FEEDBACK ON THE QOS METHOD	QUOTES
The concept of U-QASAR methodology	
Collaborative approach	Group work around the quality objectives. Good to listen to somebody else's opinion and trying to cluster together similar ideas.
Alignment stakeholders' understanding of quality	And I think that the workshop is very, very nice, to somehow align that. To align the view.
Communication	Communication, team-based elaboration of quality model, etc. [*worked well in the workshop.*]
Multi-stakeholder approach	See different expectations and points of view from different roles is great!
Concept: the method generally	It is a good exercise to think on what to measure on a high level and then to figure out how it could be measured. I like the method.
Concept: focusing the model by voting	Also voting for the objectives was a nice way to set the focus on a specific amount of objectives.
Concept: openness of the method	The open concept of the workshop, enabling a free interpretation of what model, QO, QI and metrics are.
Concept: elaboration by iterations	The iterative building of indicators and metrics
Facilitation	
Well organized	Each step worked well
Not too many participants	The work in the subgroups went very well, it is much easier to find a consensus than in a bigger group.
Content	
Elaborating the results	It was very, very useful to speak and to write down things. -- we both had a completely different understanding from this and we came to it as we spoke about it.
Organizing objectives	The clustering after the brain storming
Setting context by pre-questionnaire	Good to have made the pre-questionnaire in advance.

NEGATIVE FEEDBACK ON THE QOS METHOD	QUOTES
Facilitation	
Lack of time	The workshop could be even longer to reach more extensive results
Lack of guidance: examples of QO/QI/QM	Better guidance: As a non-expert I couldn't think of relevant objectives easily. It would be nice to give a comprehensive set of QO areas.
Lack of guidance: training on terminology	We had different understanding of what they are like the model the objective and the indicator and the metric, then what came up when we were putting the things there, they were on completely different level.
Poor structure of guidelines	Maybe, a clearer or better-structured guideline
Content and scope	
Scope: too wide	Maybe make it more focused. because now we have to go through the whole process and product. --- so we have this global meeting for the quality model.. it's too much.
Scope: too low-level	Too low-level: Lots of time used to think about metrics and the values for the metrics in the objective template.
Documenting	Documenting the output of the workshop in some online form already during the actual sessions.

LESSONS LEARNED	QUOTES
Defining the quality model	
It is difficult to decide on the limits for good and bad	But the problem is a little bit this what Tanya and me tried to do create this objective "Amount of changes in a feature" so is one change in a feature, is it good or bad?
The different parts should have different weights on the results	And there are issues here that we haven't defined, for example the weights. I think it's very important. because right now we are considering everything has the same weight but it's not.
Complex metrics need time to be elaborated	And there are issues here that we haven't defined, for example the weights. I think it's very important. because right now we are considering everything has the same weight but it's not.
Participants in defining a quality model	
Should be initially done by few people with good technical knowledge	I think it's a good starting point to get, let's say, decision makers, experts, project manager, whatever in the first stage and then start refining that maybe in different iterations.
All stakeholders should have an influence on the model	Everybody should have a say in it.
Customer should not participate	Customers have needs but they usually don't have them compressed into a level where you can get a metric out of it.

Appendix I Defined quality models

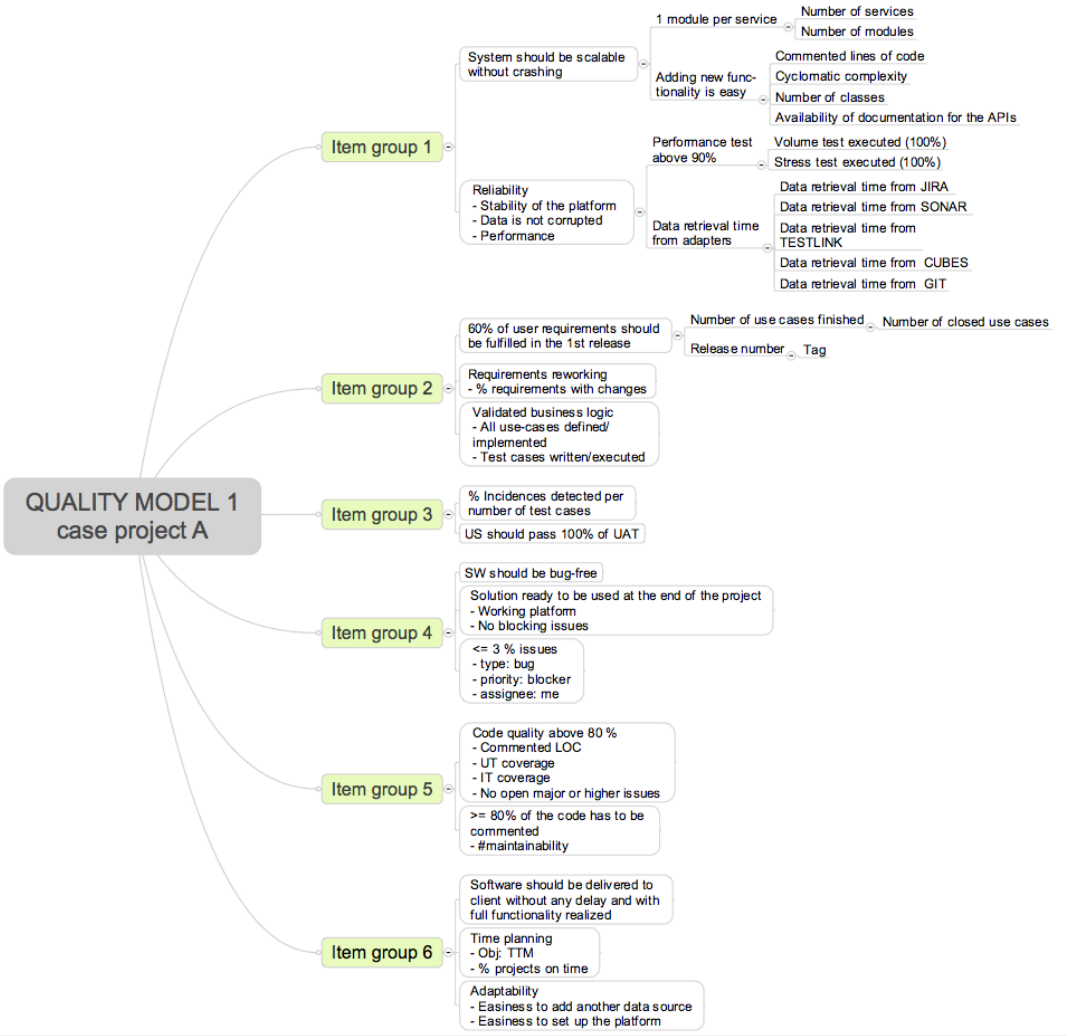

Figure 30. Quality model defined for case Project A by Group 1.

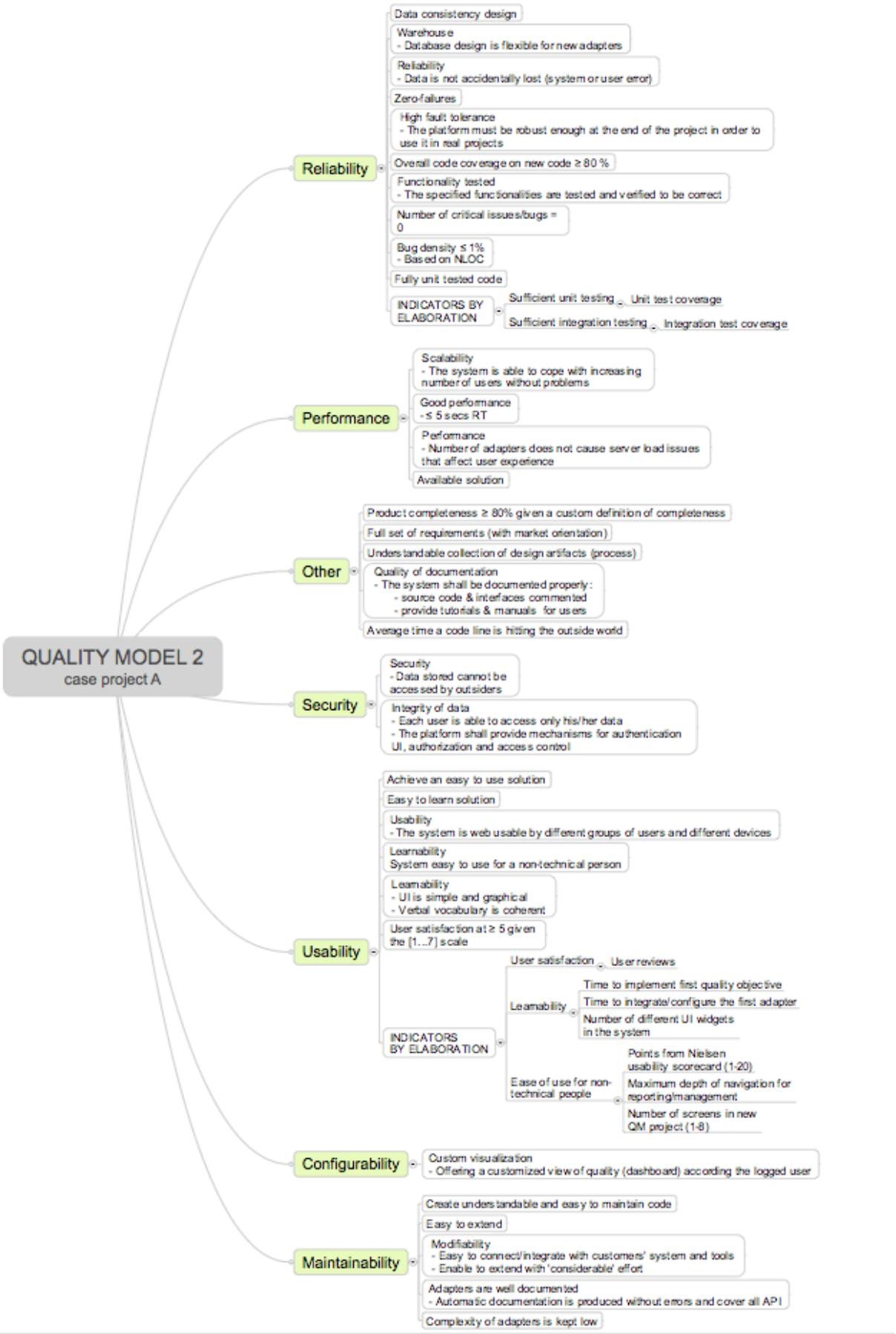

Figure 31. Quality model defined for case Project A by Group 2.

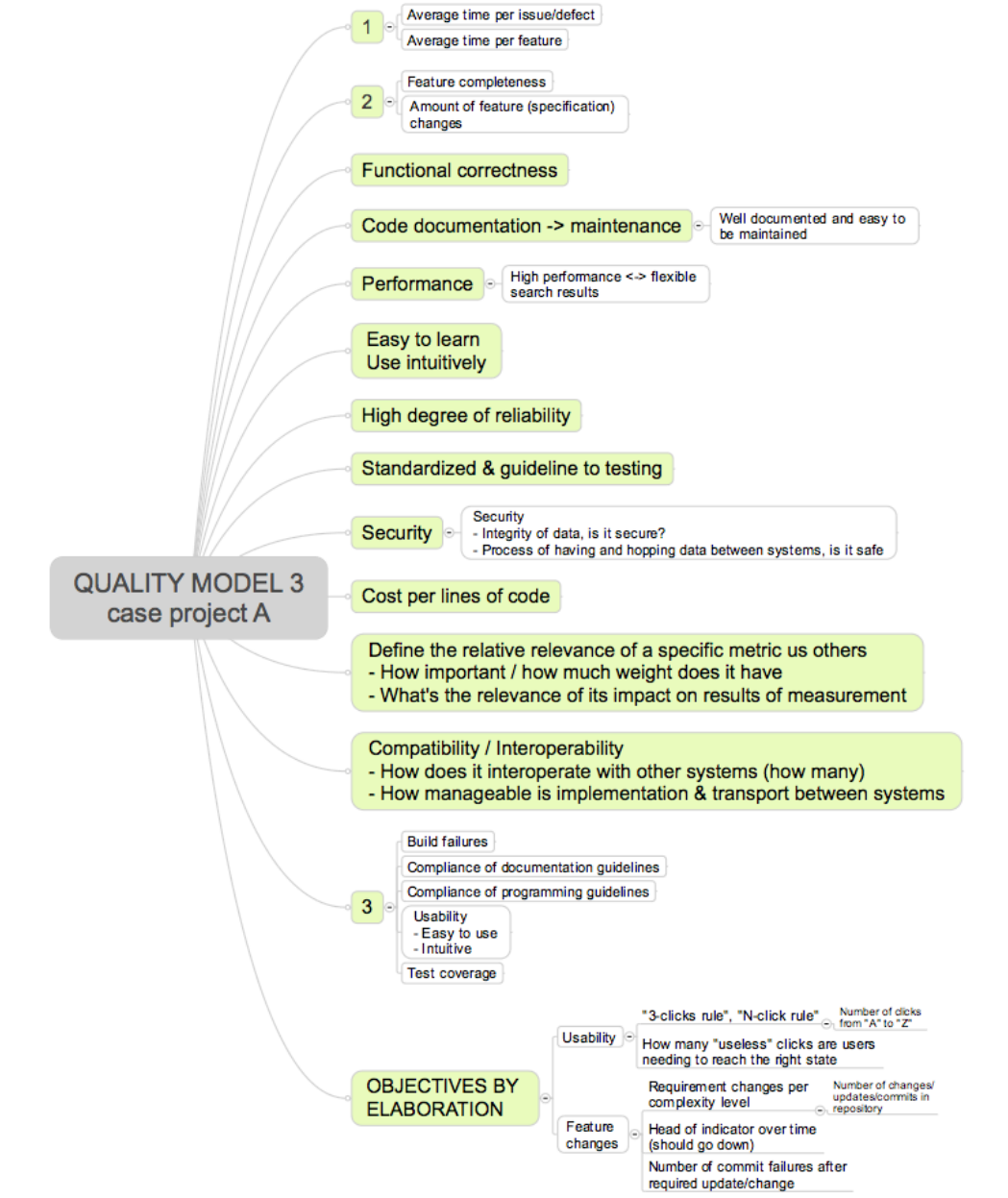

Figure 32. Quality model defined for case Project A by Group 3.

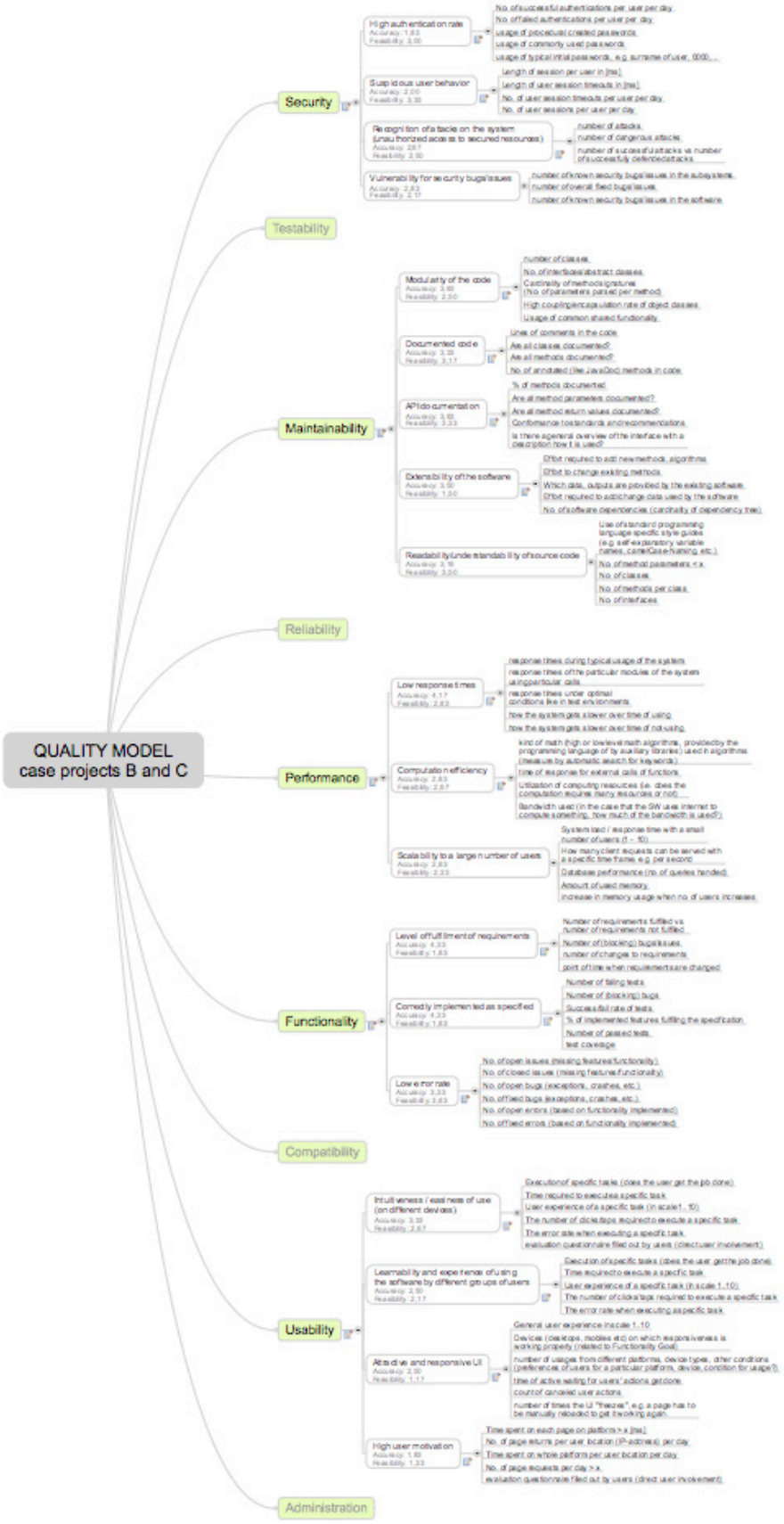

Figure 33. Quality model defined for Project B and Project C.

Appendix J Deployed quality monitoring programs

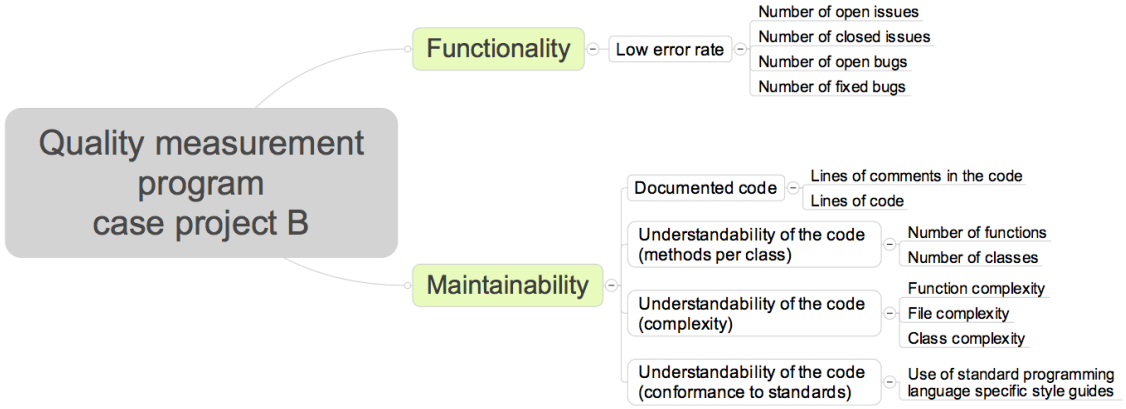

Figure 34. Quality monitoring program deployed in Project A.

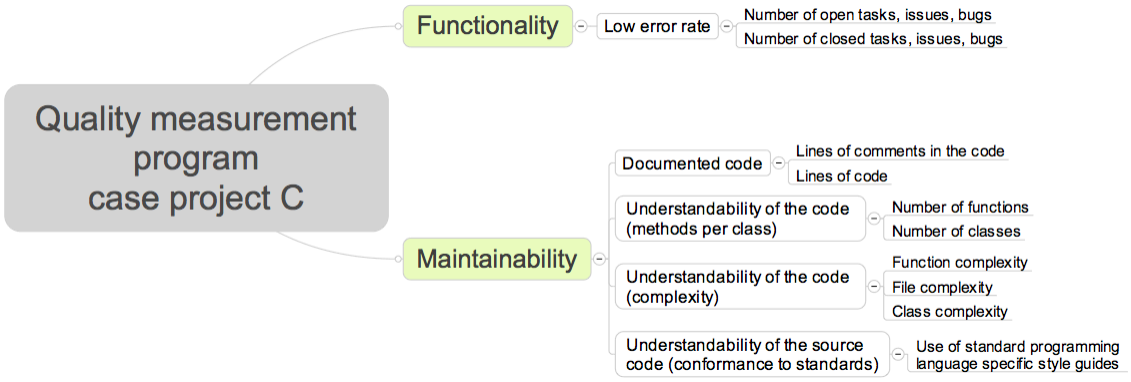

Figure 35. Quality monitoring program deployed in Project A.

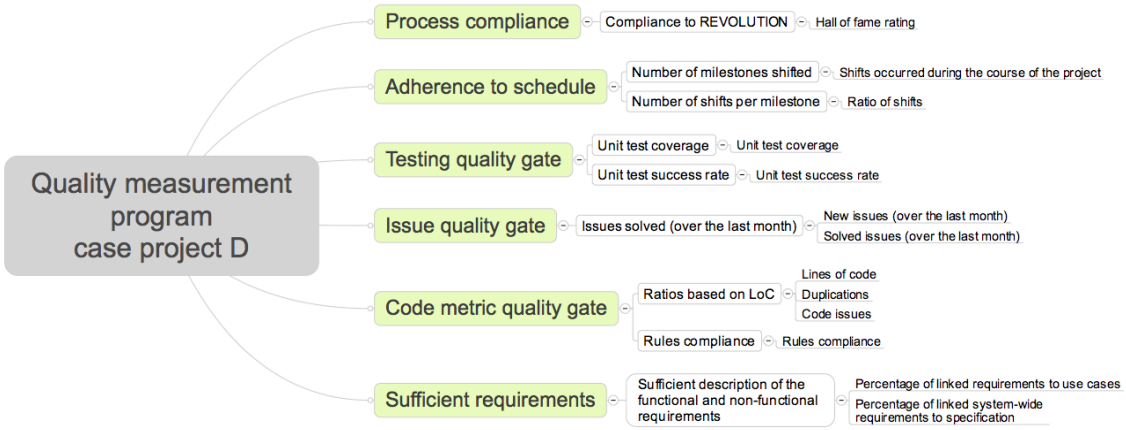

Figure 36. Quality monitoring program deployed in Project D.

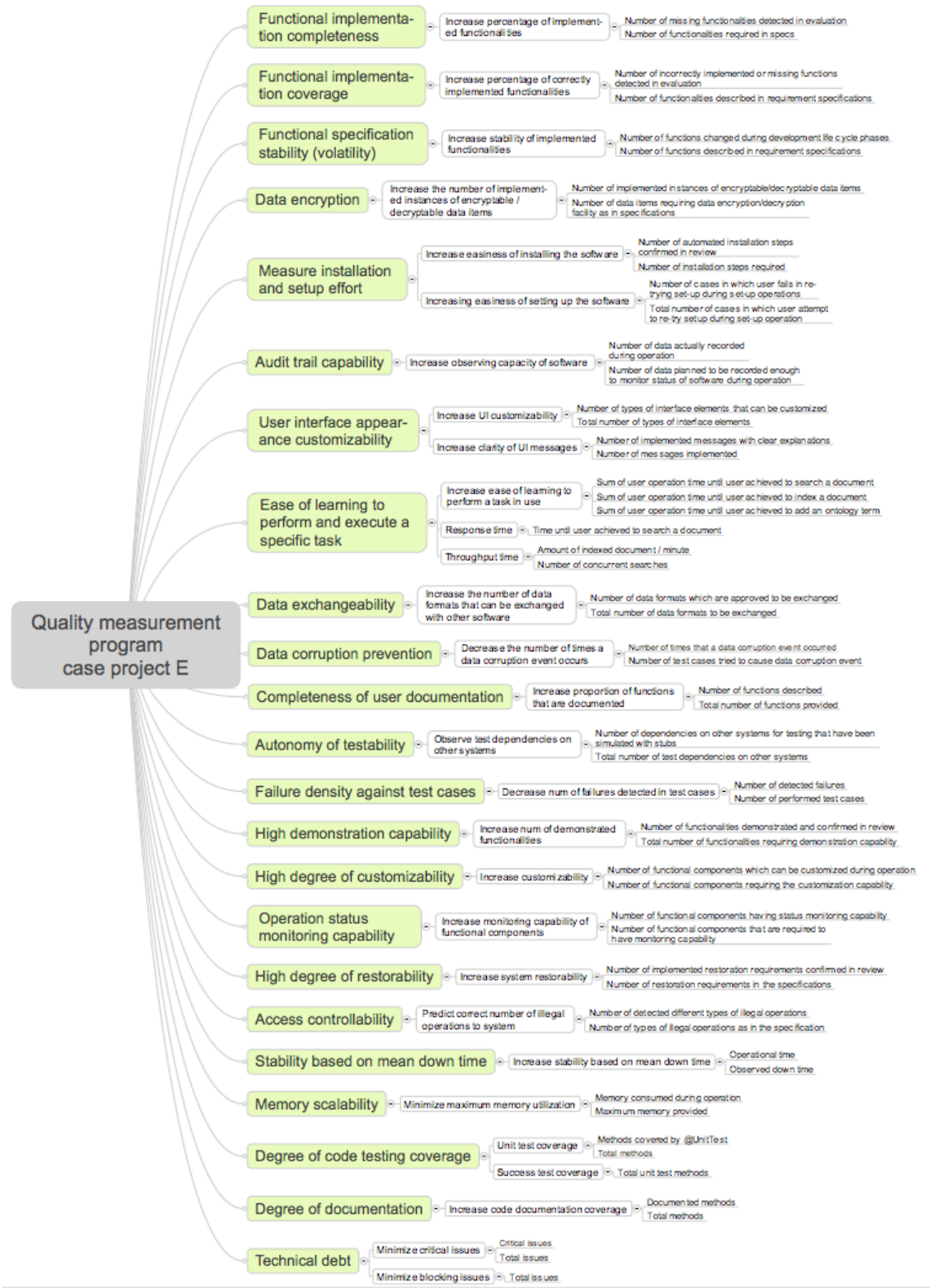

Figure 37. Quality monitoring program deployed in Project E.

Table 25. The relationship between the defined quality model and deployed quality monitoring programs: elements used from the quality model in the quality monitoring program in Project B and Project C.

Defined model			**Measurement program**	
Objectives	**Indicators**	**Metrics**	**Project B**	**Project C**
Functionality	Level of fulfillment of requirements	Number of requirements fulfilled vs. number of requirements not fulfilled		
		Number of (blocking) bugs/issues		
		Number of changes to requirements		
		Point of time when requirements are changed		
	Correctly implemented as specified	Number of failing tests		
		Number of (blocking) bugs		
		Success/fail rate of tests		
		% of implemented features fulfilling the specification		
		Number of passed tests		
		Test coverage		
	Low error rate	No. of open issues (missing features/functionality)	x	x
		No. of closed issues (missing features/functionality)	x	x
		No. of open bugs (exceptions, crashes, etc.)	x	x
		No. of fixed bugs (exceptions, crashes, etc.)	x	x
		No. of open errors (based on functionality implemented)		
		No. of fixed errors (based on functionality implemented)		
Maintainability	Modularity of the code	Number of classes		
		No. of interfaces/abstract classes		
		Cardinality of method signatures		
		(No. of parameters parsed per method)		
		High coupling/encapsulation rate of object classes		
		Usage of common shared functionality		
	Documented code	Lines of comments in the code	x	x
		Are all classes documented?		
		Are all methods documented?		
		No. of annotated (like JavaDoc) methods in code		
	API documentation	% of methods documented		
		Are all method parameters documented?		
		Are all method return values documented?		
		Conformance to standards and recommendations		
		Is there a general overview of the interface with a description how it is used?		
	Extensibility of the software	Effort required to add new methods, algorithms		
		Effort to change existing methods		
		Which data, outputs are provided by the existing software		
		Effort required to add/change data used by the software		
		No. of software dependencies (cardinality of dependency tree)		
	Readability/ understandability of source code	Use of standard programming language specific style guides		
		No. of method parameters < x No. of classes		
		No. of methods per class		
		No. of interfaces		

Appendix K Quotes on quality monitoring

Table 26. Quotes behind the quality monitoring feedback categories.

FEEDBACK ON THE USEFULNESS OF QUALITY MONITORING WITH THE U-QASAR PLATFORM	QUOTES
Usefulness of data integration	
Saving time and effort by having all data in one place	Personally, I think that the greatest benefit of the platform could be the amount of time saved by not having to look for any kind of information in various systems
Saving time and effort by providing simplified data	To have a quite quick overview of the quality of the system based on this graphics. You get a pretty quick expression of the overall situation of the project.
Usefulness of the collected quality information	
Creating awareness of the project state	That would be one thing, to see the development over time.
Providing tangible proof of success	For me, it helps me provide tangible proof of a successful process execution.
Justifying & facilitating decision-making	uQasar could be the means of realizing prescriptive analytics i.e. it could be the tool that would justify the need for a specific decision
Justifying & facilitating process improvement	But we are confident that this [*improving processes*] could have been done in a more active time of development activities.
Justifying & facilitating resource management	For the project manager i think that the most valuable feature is that it provides him the ability to see if a project execution is performed with the correct trade-offs.
Providing assurance of right development actions	It is possible to get a quick overview of whether the current software development is on the “right way” towards a good software quality (achieving the defined objectives) or not
Generate communication	For example like that a serious decline in quality from one week to the next could stay definitely a point to talk about.
Usefulness of different approaches of presenting the data	
Providing adequate use for stakeholders by customization	Possibility for different “stakeholders” (managers, developers, testers, etc.) to use the system adequately
Providing more abstract information	Maybe our boss could take look and have an overview of all the ...--.. projects running and
Providing information on different detail-levels	While there is general information about the quality model and its criteria there are several levels going more into detail. Therefore this overview is able to serve as abstract and as detailed information source, e.g. for management activities and also for development and implementation activities

FEEDBACK ON THE CHALLENGES OF QUALITY MONITORING WITH THE U-QASAR PLATFORM	QUOTES
Challenges of interpreting the data	
Interpreting the quality objectives/indicators/metrics	Getting the real semantics of the gathered metrics like e.g. what does the Complexity metric mean exactly
Interpreting the results provided by the platform	My main question would be how do the numbers that are being produced by UQASAR help me to make the code better // Yes, are there some tips and guidelines to tell me that this is a good value, this is not a good value?
Understanding the data collection and calculations	It wasn't clear how all the things were being collected.
Understanding the graphics of the platform	The purpose of some UI design elements was not immediately clear to the users. Suggestion for improvement were communicated to the U-QASAR team.
User challenges	
Motivation	Keeping the motivation high to use U-QASAR as an integrated part of the own daily/weekly work
Phase and size of the project	UQASAR has not been used because there has not been any development and the metrics that are used are related to the code only
Time used for manual metrics	The challenge of using U-QASAR at the moment and in our case specifically was the time you’d have to spend entering all the manually gathered metrics.
Change resistance & costs	It's yet another tool that needs to be integrated to the tooling landscape so.. that might be problematic. It might cause overhead there.. In the administration part

Appendix L Quotes on the adapter development

Table 27. Quotes behind the challenges and restrictions in adapter development.

Aspect	Findings	Quotes
Challenges	Privacy of the individual employees working for the institution	Intrasoft would like to combine the version management information to other information, so there's additionally the privacy matters
	Varying configuration settings of the target system	Yeah, because the TestLink tool is not so flexible so everything needed to be well configured in order to have a response from the API. If not, you got an error but you didn't know what happened.
		One of the problems is that for having these metrics you have to select test project and the test plan. If you don't have everything configured in the TestLink, the API methods fail.
	No API available, connection straight to the database	No, because the TestLink adapter is too restricted so we can't do it more with this API. Maybe we have to use not the API for TestLink instead of programming a completely new adapter with queries directly to the database.
		We could use the same TestLink adapter but adding new methods which query directly to the database. And squeeze (??) them up with other methods.
	Authentication challenges	And at the moment the information security or the authentication could be done in some other way, like now we are still using the usernames and passwords. Maybe there could be some other method to use that would be more flexible and maybe also more secure. So we would not need to save the passwords.
	Negligence on understanding the implementation	On the other hand, when we use each other's code we might make the same mistakes and might not think things through. When trying to ready quickly..
Restrictions	The metric data format is restricted	Well, of course the point that the data should be saved in a certain format generated some additional work. All the adapter data is saved as "measurement objects" to the database.
		I think it's also very important to have in mind how the results can be presented. Are you going to retrieve a number, are you going to retrieve a series of numbers, a file of numbers..
		It has a JSON file, [...] so you have to put the information on the right field. This is the main constraint.
		The only important thing is.. well it has to be in some format which is transformable into the target format. Part of the interface is, that these train supply form consume are JSON formatted and have to be of some shape. The tree has to be transformable into that. That's the only requirement.
	Generality of the data that can be retrieved	And also that it is easy to just, like in many things concerning the platform, not to think enough about the generic implementation but developing some independent things for each system. It makes the system more difficult to maintain and on the other hand, also mistakes are made more easily.
	Data processing in U-QASAR platform is constrained	The restrictions of the platform can maybe be seen in that way that a lot of changes has to be made in the platform side depending on what is wanted to be done with the data. So saving the data is a piece of a cake, but the further processing of it and the use in the metrics is quite difficult right now
		it is quite restricted by now what you can do on the platform side. After the retrieval of the data.
	The account rights to the target system might be a problem	Well basically the adapter has the rights that are put for the user account that the adapter uses. And of course to what the API enables.
		So actually it would be sensible to create a separate user account for the adapter.
		And at the moment the information security or the authentication could be done in some other way, like now we are still using the usernames and passwords. Maybe there could be some other method to use that would be more flexible and maybe also more secure. So we would not need to save the passwords.
	The U-QASAR platform is restricted so that it only accepts refined data that can directly be used in the metrics	It would be better to keep the platform side quite generic so that there would not have to be done so much work and the metrics would be gotten directly in the correct format from the adapter.
		Well the computation stuff has been done on the platform side until this because the adapters are in that way dummy that they only collect the data that is available at that moment. If they would use data that was saved in a database, it would enable some other things.
		Yeah, and also the direction of the development of the common interface of the platform should be discussed.
		it is quite restricted by now what you can do on the platform side. After the retrieval of the data.
		But I think that if this is going to be developed as an open source project, this is one a matter that has to be developed further. The adapter interface
		Well at the moment it depends on the common interface, it just has to be implemented like that. It is tied to the platform
		And also that it is easy to just, like in many things concerning the platform, not to think enough about the generic implementation but developing some independent things for each system. It makes the system more difficult to maintain and on the other hand, also mistakes are made more easily.
		I think that it's not a problem to know nothing about the platform.
		I was familiar with the most important parts when you're implementing and adapter with the interface. But not with the rest of the UQASAR code.
		The thing with this issue is that the model which we have implemented from the start on, was.. we do it all at the project level and all the metrics are related to the project itself. We never had a concept of splitting up the project into ingredients and there was also no notion about issues for developers and measuring them. So there is just no concept how to do this in our design.
		Because for bigger projects it is certainly useful to look at subproject level like components and track the metrics at that level. But, as I already said, to be able to do that and to be able to do that cross all adapters, the interface and the platform have to be extended so it knows the components of the project.

Appendix M Source code of the Jenkins adapter

```
public class JenkinsAdapter implements SystemAdapter {

/************** QUERY METHOD ADAPTER - JENKINS  **************/
@Override
public List<Measurement> query(BindedSystem boundSystem, User user,
        QueryExpression queryExpression) throws uQasarException {

/* A linked list variable for storing the measurements */
LinkedList<Measurement> measurements = new LinkedList<Measurement>();
try {
        /* Saving the system url */
        String long_url = boundSystem.getUri();
        String url = "";
        /* Saving the project name */
        String project = "";
        /* Testing weather the url is valid and removing the end of it to get an url for authentication */
        String[] parts = long_url.split("/");

        //***** Splitting the URL to URL & project name ***** //
        for(int i = 0; i < parts.length; i++) {
                if(parts[i].compareTo("job") == 0 && parts.length-1 > i){
                        project = parts[i+1];
                        i = parts.length; }
                        else if (i == 3){
                                url = url + parts[i]; }
                        else{url = url + parts[i] + "/";}}

        /* Initialize new Jenkins server */
        JenkinsServer jenkins = null;
        /* Use the Jenkins-client library to open the server connection */
        //***** Connecting to server ***** //
        jenkins = new JenkinsServer(new URI(url), user.getUsername(), user.getPassword());
        /* This comes from the U-QASAR interface */
        String query = queryExpression.getQuery();

        //***** Implementing data mapping *****//
        if (query.equalsIgnoreCase(uQasarMetric.JENKINS_LATEST_BUILD_SUCCESS.name())) {
                // Here the status of the latest build is fetched
                // Only status "Stable", "Unstable" and "Failed" are noticed.
                JobWithDetails job  = null;
                String status = "";
                job = jenkins.getJobs().get(project).details();
                if(job.getLastBuild().getNumber() == job.getLastStableBuild().getNumber()){
                        status = "Stable";}
                else if (job.getLastBuild().getNumber() == job.getLastUnstableBuild().getNumber()) {
                        status = "Unstable";}
                else if (job.getLastBuild().getNumber() == job.getLastFailedBuild().getNumber()) {
                        status = "Broken";}
                else
                        status = "Unknown";
                measurements.add(new
                Measurement(uQasarMetric.JENKINS_LATEST_BUILD_SUCCESS, status)); }
        if (query.equalsIgnoreCase(uQasarMetric.JENKINS_BUILD_HISTORY.name())) {
                // Here the status of the maximum 100 latest builds are fetched
```

```
                    // Only status "Stable", "Unstable" and "Failed" are noticed.
                    JobWithDetails job  = null;
                    JSONArray measurementResultJSONArray = new JSONArray();
                    job = jenkins.getJobs().get(project).details();

                    for(int i=0; i< Math.min(100, job.getBuilds().size()); i++ ) {
                            JSONObject jObj = new JSONObject();
                            jObj.put("BuildNumber",
                            Integer.toString(job.getBuilds().get(i).details().getNumber()));

                            if(job.getBuilds().get(i).details().getResult().name() != "STABLE") {
                                                    jObj.put("BuildStatus", "Stable");}
                            else if(job.getBuilds().get(i).details().getResult().name() != "UNSTABLE") {
                                                    jObj.put("BuildStatus",
                            "Unstable");}
                            else if(job.getBuilds().get(i).details().getResult().name() != "FAILED") {
                                                    jObj.put("BuildStatus",
                            "Broken");}
                            else {jObj.put("BuildStatus", "Other");}

                            measurementResultJSONArray.put(jObj); }

                    measurements.add(new Measurement(uQasarMetric.JENKINS_BUILD_HISTORY,
                    measurementResultJSONArray.toString()));
            }

            if (query.equalsIgnoreCase(uQasarMetric.JENKINS_PROJECTS.name())) {
                    // Here all the projects in the Jenkins instance are fetched
                    // Name, url, last build.
                    JobWithDetails job  = null;
                    Map<String, Job> jobs = null;
                    JSONArray measurementResultJSONArray = new JSONArray();
                    job = jenkins.getJobs().get(project).details();
                    jobs = jenkins.getJobs();
                    for (Map.Entry entry : jobs.entrySet()) {
                            JSONObject jObj = new JSONObject();
                            Job j = (Job) entry.getValue();
                            jObj.put("name", j.getName());
                            jObj.put("url", j.getUrl());
                            if(j.details().getLastBuild() != null) {
                                    jObj.put("last_build",
            Integer.toString(j.details().getLastBuild().getNumber()));
                            }
                            else {
                                    jObj.put("last_build", "no_builds");}
                            measurementResultJSONArray.put(jObj);}

                    measurements.add(new Measurement(uQasarMetric.JENKINS_PROJECTS,
                    measurementResultJSONArray.toString()));
            }
            return measurements;
    } catch (Exception e) {
            e.printStackTrace();
            return measurements; }
    }
```